SKOS Simple Knowledge Organization System Primer

W3C Working Group Note 18 August 2009

This version:
http://www.w3.org/TR/2009/NOTE-skos-primer-20090818/
Latest version:
http://www.w3.org/TR/skos-primer
Previous version:
http://www.w3.org/TR/2009/WD-skos-primer-20090615/
Editors:
Antoine Isaac, Vrije Universiteit Amsterdam
Ed Summers, Library Of Congress

Please refer to the **errata** for this document, which may include some corrections.

See also **translations**.

Abstract

SKOS—Simple Knowledge Organization System—provides a model for expressing the basic structure and content of concept schemes such as thesauri, classification schemes, subject heading lists, taxonomies, folksonomies, and other similar types of controlled vocabulary. As an application of the Resource Description Framework (RDF), SKOS allows concepts to be composed and published on the World Wide Web, linked with data on the Web and integrated into other concept schemes.

This document is a user guide for those who would like to represent their concept scheme using SKOS.

In basic SKOS, conceptual resources (concepts) are identified with URIs, labeled with strings in one or more natural languages, documented with various types of note, semantically related to each other in informal hierarchies and association networks, and aggregated into concept schemes.

In advanced SKOS, conceptual resources can be mapped across concept schemes and grouped into labeled or ordered collections. Relationships can be specified between concept labels. Finally, the SKOS vocabulary itself can be extended to suit the needs of particular communities of practice or combined with other modeling vocabularies.

This document is a companion to the SKOS Reference, which provides the normative reference on SKOS.

Status of this Document

This section describes the status of this document at the time of its publication. Other documents may supersede this document. A list of current W3C publications and the latest revision of this technical report can be found in the W3C technical reports index at http://www.w3.org/TR/.

This document is a Working Group Note published by the Semantic Web Deployment Working Group, part of the W3C Semantic Web Activity. This version is an update to the previous Working Draft of 15 June 2009. This version includes several minor editorial changes as well as removing an example that suggested one means to reference a system of notation (e.g. a symbolic notation) in a label where the system of notation does not correspond to a natural language. This suggestion was deemed inconsistent with IETF Best Current Practice 47 on the use of tags for identifying languages. Users should consider the SKOS Extension vocabulary for support of alternate systems of notation.

This is a companion document to the SKOS Simple Knowledge Organization System Reference W3C Recommendation dated 18 August 2009.

Comments on this document may be sent to public-swd-wg@w3.org; please include the text "SKOS comment" in the subject line. All messages received at this address are viewable in a public archive.

This document was produced by a group operating under the 5 February 2004 W3C Patent Policy. W3C maintains a public list of any patent disclosures made in connection with the deliverables of the group; that page also includes instructions for disclosing a patent. An individual who has actual knowledge of a patent which the individual believes contains Essential Claim(s) must disclose the information in accordance with section 6 of the W3C Patent Policy.

Publication as a Working Group Note does not imply endorsement by the W3C Membership. This is a draft document and may be updated, replaced or obsoleted by other documents at any time. It is inappropriate to cite this document as other than work in progress.

Table of Contents

---

1 Introduction

The Simple Knowledge Organization System (SKOS) is an RDF vocabulary for representing semi-formal *knowledge organization systems* (KOSs), such as thesauri, taxonomies, classification schemes and subject heading lists. Because SKOS is based on the Resource Description Framework (RDF) [*RDF-PRIMER*] these representations are machine-readable and can be exchanged between software applications and published on the World Wide Web.

SKOS has been designed to provide a low-cost migration path for porting existing organization systems to the Semantic Web. SKOS also provides a lightweight, intuitive conceptual modeling language for developing and sharing new KOSs. It can be used on its own, or in combination with more-formal

languages such as the Web Ontology Language (OWL) [*OWL*]. SKOS can also be seen as a bridging technology, providing the missing link between the rigorous logical formalism of ontology languages such as OWL and the chaotic, informal and weakly-structured world of Web-based collaboration tools, as exemplified by social tagging applications.

The aim of SKOS is not to replace original conceptual vocabularies in their initial context of use, but to allow them to be ported to a shared space, based on a simplified model, enabling wider re-use and better interoperability.

1.1 About this Primer

This document is intended to help users who have a basic understanding of RDF to represent and publish their concept schemes as SKOS data. The Primer aims to provide introductory examples and guidance in the use of the SKOS vocabulary.

For a systematic account of all SKOS vocabulary elements, including their reference semantics, the reader should consult the normative *SKOS Reference* [*SKOS-REFERENCE*]. This can be done, at the level of classes and properties, by clicking on their occurrences in the text (e.g. skos:Concept). For an overview of the use cases for SKOS and the elicited requirements that guided its design, the reader should consult the *SKOS Use Cases and Requirements* [*SKOS-UCR*].

This Primer, together with the *SKOS Reference* [*SKOS-REFERENCE*], replaces the earlier *SKOS Core Guide* [*SWBP-SKOS-CORE-GUIDE*] and the *SKOS Core Vocabulary Specification* [*SWBP-SKOS-CORE-SPEC*], which are now deprecated.

The essential features of the SKOS model are explained in Section 2. Here, the reader is presented with the set of vocabulary elements that are most commonly used for representing KOSs. In Section 3, the reader is shown how to add value to these representations, either by linking them together or by relating them to other kinds of Semantic Web resources. It is expected that many SKOS applications will employ some of the features presented in Section 3. Section 4 is focused on more-advanced representation needs, which are likely to be required for a limited number of SKOS applications. Section 5 discusses the use of SKOS in conjunction with other modeling approaches, specifically OWL.

About Examples in this Primer

Most of the examples in this guide are given as a serialization of RDF graphs using the Turtle syntax for RDF [*TURTLE*]. Examples serialized as Turtle appear in code lines such as:

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix ex: <http://www.example.com/>.

ex:aResource ex:aProperty ex:anotherResource;
```

```
ex:anotherProperty "An RDF Literal"@en.
```

The above is equivalent to the following expression, in the RDF/XML reference syntax [*RDF/XML-SYNTAX*]:

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:ex="http://www.example.com/">
<rdf:Description rdf:about="http://www.example.com/aResource">
  <ex:aProperty rdf:resource="http://www.example.com/anotherResource"
  <ex:anotherProperty xml:lang="en">An RDF Literal</ex:anotherPropert
</rdf:Description>
</rdf:RDF>
```

For the sake of brevity a number of namespace declarations are omitted from the examples. This applies to standard namespaces (SKOS, RDF/RDFS [*RDF-PRIMER*], OWL [*OWL*] and Dublin Core [*DC*]) but also to the ones that are coined for the examples. Generally, these namespaces could be declared as in the following code:

```
@prefix skos: <http://www.w3.org/2004/02/skos/core#> .
@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 dct: <http://purl.org/dc/terms/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix ex: <http://www.example.com/> .
@prefix ex1: <http://www.example.com/1/> .
@prefix ex2: <http://www.example.com/2/> .
```

---

2 SKOS Essentials

This section introduces the core of the SKOS model, namely the features that are needed to represent most KOSs, as observed in the majority of use cases [*SKOS-UCR*].

In basic SKOS, conceptual resources (concepts) can be identified with URIs, labeled with lexical strings in one or more natural languages, documented with various types of note, semantically related to each other in informal hierarchies and association networks and aggregated into concept schemes.

2.1 Concepts

The fundamental element of the SKOS vocabulary is the *concept*. Concepts are the *units of thought* [*WillpowerGlossary*]—ideas, meanings, or (categories of) objects and events—which underlie many knowledge organization systems [*SKOS-UCR*]. As such, concepts exist in the mind as abstract entities which are independent of the terms used to label them.

SKOS introduces the class `skos:Concept`, which allows implementors to assert that a given resource is a concept. This is done in two steps:

1. by creating (or reusing) a Uniform Resource Identifier (URI [*URI*]) to uniquely identify the concept.
2. by asserting in RDF, using the property `rdf:type`, that the resource identified by this URI is of type `skos:Concept`.

For example:

```
<http://www.example.com/animals> rdf:type skos:Concept.
```

This can also be represented in Turtle more compactly using the namespace prefix `ex` defined above:

```
ex:animals rdf:type skos:Concept.
```

Using SKOS to publish concept schemes makes it easy to reference the concepts in resource descriptions on the Semantic Web. Implementors are encouraged to use HTTP URIs when minting concept URIs since they are resolvable to representations that can be accessed using standard Web technologies. For more information about URIs on the Semantic Web, see *Cool URIs for the Semantic Web* [*COOLURIS*] and *Best Practice Recipes for Publishing RDF Vocabularies* [*RECIPES*].

2.2 Labels

The first characterizations of concepts are the expressions that are used to refer to them in natural language: their *labels*. SKOS provides three properties to attach labels to conceptual resources: `skos:prefLabel`, `skos:altLabel` and `skos:hiddenLabel`. Each property implies a specific status for the label it introduces, ranging from a strong, univocal denotation relationship, to a string to aid in lookup. These properties are formally defined as being pairwise disjoint. This means, for example, that it is an error if a concept has a same literal both as its preferred label and as an alternative label.

As specified in Section 5 of the *SKOS Reference*, `skos:prefLabel`, `skos:altLabel` and `skos:hiddenLabel` provide simple labels. They are all sub-properties of `rdfs:label`, and are used to link a `skos:Concept` to an RDF plain literal, which is a character string (e.g. `"love"`) combined with an optional language tag (e.g. `"en-US"`) [*RDF-CONCEPTS*].

2.2.1 Preferred Lexical Labels

The `skos:prefLabel` property makes it possible to assign a preferred lexical label to a resource. Terms used as *descriptors* in indexing systems [*WillpowerGlossary*] will for instance be represented using this property, as in the following example:

```
ex:animals rdf:type skos:Concept;
  skos:prefLabel "animals".
```

RDF plain literals are formally defined as character strings with optional language tags. SKOS thereby enables a simple form of multilingual labeling. This is done by using the language tag of a lexical label to restrict its scope to a particular language. The following example illustrates how a concept is given one preferred label in English and another in French:

```
ex:animals rdf:type skos:Concept;
  skos:prefLabel "animals"@en;
  skos:prefLabel "animaux"@fr.
```

Note that the notion of *preferred* label implies that a resource can only have one such label per language tag, as explained in Section 5 of the *SKOS Reference* [*SKOS-REFERENCE*].

Following common practice in KOS design, the preferred label of a concept may also be used to unambiguously represent this concept within a KOS and its applications. So even though the SKOS data model does not formally enforce it, it is recommended that no two concepts in the same KOS be given the same preferred lexical label for any given language tag.

2.2.2 Alternative Lexical Labels

The `skos:altLabel` property makes it possible to assign an alternative lexical label to a concept. This is especially helpful when assigning labels beyond the one that is preferred for the concept, for instance when synonyms need to be represented:

```
ex:animals rdf:type skos:Concept;
  skos:prefLabel "animals"@en;
  skos:altLabel "creatures"@en;
  skos:prefLabel "animaux"@fr;
  skos:altLabel "créatures"@fr.
```

Note that representation of synonyms for preferred labels is not the only use for `skos:altLabel`. *Near-synonyms*, abbreviations and acronyms can be represented the same way:

```
ex:fao rdf:type skos:Concept;
  skos:prefLabel "Food and Agriculture Organization"@en;
  skos:altLabel "FAO"@en.
```

> **Note on upward posting:** It is also possible to use `skos:altLabel` to represent cases of *upward posting* [*ISO-2788*]. That is, when a concept aggregates more-specialized notions that are not explicitly introduced as concepts in the considered KOS:
>
> ```
> ex:rocks rdf:type skos:Concept;
>   skos:prefLabel "rocks"@en;
>   skos:altLabel "basalt"@en;
>   skos:altLabel "granite"@en;
> ```

```
skos:altLabel "slate"@en.
```

However, even though SKOS is not intended to replace existing guides for KOS design [*ISO-2788*, *BS8723-2*], the reader should be aware that upward posting is not recommended. A more appropriate KOS for this domain would introduce a `skos:Concept` for each kind of rock considered (basalt, granite and slate) and assert it as a narrower concept of `ex:rock`.

2.2.3 Hidden Lexical Labels

A *hidden lexical label*, represented by means of the `skos:hiddenLabel` property, is a lexical label for a resource, where a KOS designer would like that character string to be accessible to applications performing text-based indexing and search operations, but would **not** like that label to be visible otherwise. Hidden labels may for instance be used to include misspelled variants of other lexical labels. For example:

```
ex:animals rdf:type skos:Concept;
  skos:prefLabel "animaux"@fr;
  skos:altLabel "bêtes"@fr;
  skos:hiddenLabel "betes"@fr.
```

2.3 Semantic Relationships

In KOSs *semantic relations* play a crucial role for defining concepts. The meaning of a concept is defined not just by the natural-language words in its labels but also by its links to other concepts in the vocabulary. Mirroring the fundamental categories of relations that are used in vocabularies such as thesauri [*ISO2788*], SKOS supplies three standard properties:

- `skos:broader` and `skos:narrower` enable the representation of hierarchical links, such as the relationship between one *genre* and its more specific *species*, or, depending on interpretations, the relationship between one *whole* and its *parts*;
- `skos:related` enables the representation of associative (non-hierarchical) links, such as the relationship between one type of *event* and a category of entities which typically *participate* in it. Another use for `skos:related` is between two categories where neither is more general or more specific. Note that `skos:related` enables the representation of associative (non-hierarchical) links, which can also be used to represent part-whole links that are not meant as hierarchical relationships.

2.3.1 Broader/Narrower Relationships

To assert that one concept is broader in meaning (i.e. more general) than another, the `skos:broader` property is used. The `skos:narrower` property is used to assert the inverse, namely when one concept is narrower in meaning (i.e. more specific) than another. For example:

```
ex:animals rdf:type skos:Concept;
  skos:prefLabel "animals"@en;
  skos:narrower ex:mammals.
ex:mammals rdf:type skos:Concept;
  skos:prefLabel "mammals"@en;
  skos:broader ex:animals.
```

As is often the case in KOS, a SKOS concept can be attached to several broader concepts at the same time. For example, a concept `ex:dog` could have both `ex:mammals` and `ex:domesticatedAnimals` as broader concepts.

> **Note on `skos:broader` direction:** for historic reasons, the name of the `skos:broader` property (the word "broader") does not provide an explicit indication of its direction. The word "broader" should read here as "has broader concept"; the subject of a `skos:broader` statement is the more specific concept involved in the assertion and its object is the more generic one.

> **Note on implicit `skos:broader`/`skos:narrower` statements**: the properties `skos:broader` and `skos:narrower` are each other's inverse. Whenever a concept X is broader than another concept Y, then Y is a narrower concept of X according to the SKOS data model [*SKOS-REFERENCE*]. This can be useful for making SKOS representations more efficient by limiting the information they contain. In the above example, for instance, the statement `ex:mammals skos:broader ex:animals` can be left out if, before using the concept scheme data, an OWL reasoner is used to infer it from the statement `ex:animals skos:narrower ex:mammals`.

In many cases, hierarchical relations in a concept scheme can be considered as transitive [*OWL*]. If `ex:animals` is broader than `ex:mammals`, which is itself broader than `ex:cats`, it makes sense to assert that `ex:animals` is broader than `ex:cats`. However, there are "dirtier" hierarchies, especially in KOSs different from standard well-designed thesauri, where such a feature would not be judged appropriate. Consider for instance a case where `ex2:vehicles` is said to be broader than `ex2:cars`, which is itself asserted to be broader than `ex2:wheels`. It may be problematic if "wheels" is automatically inferred to be a narrower concept with respect to "vehicles". SKOS anticipates such problems by **not** defining `skos:broader` and `skos:narrower` as generally transitive properties. The reader interested in representing "transitive hierarchies" is encouraged to read Section 4.5, which presents a way to do this while retaining compatibility with the semantics of `skos:broader` defined in this section.

> **Note on not transitive vs. intransitive**: the SKOS model does not state that `skos:broader` and `skos:narrower` are transitive. Yet this does not imply that these properties are *intransitive*. Consider a concept `cats` which is narrower than a concept `mammals`, itself narrower than `animals`: one can assert that `cats` is narrower than `animals` as well, while staying compatible with the SKOS model. Not specifying `skos:broader` as transitive implies that no new `skos:broader` statement can be *inferred* between `cats` and `animals` by

applying SKOS axioms. This does **not** prevent the publishers of a SKOS concept scheme from *asserting* hierarchical statements that reflect a locally transitive behaviour.

Similarly, SKOS does not assume that hierarchical relations are by default irreflexive. In many thesaurus guidelines, it is prohibited to have a concept broader than itself. However, in specific cases beyond classical thesauri, some reflexive `skos:broader` statements may occur. Consider the conversion of an existing RDFS/OWL ontology into a SKOS concept scheme. In such a case, it is legitimate that every `rdfs:subClassOf` statement will be re-interpreted as a `skos:broader` statement. However, `rdfs:subClassOf` is a reflexive property, which means that for every class `C`, the statement `C rdfs:subClassOf C` is true [*OWL*]. In this case every concept would therefore have itself among its broader concepts.

Not covered in basic SKOS is the distinction between types of hierarchical relation: for example, instance-class and part-whole relationships. The interested reader is referred to Section 4.7, which describes how to create specializations of semantic relations to deal with this issue.

2.3.2 Associative Relationships

To assert an associative relationship between two concepts, `skos:related` can be used:

```
ex:birds rdf:type skos:Concept;
  skos:prefLabel "birds"@en;
  skos:related ex:ornithology.
ex:ornithology rdf:type skos:Concept;
  skos:prefLabel "ornithology"@en.
```

As described in the *SKOS Reference* [*SKOS-REFERENCE*], the `skos:related` property is symmetric [*OWL*]. From the above RDF graph, for instance, it follows that `ex:ornithology` is the subject of a `skos:related` statement that has `ex:birds` as an object.

> **Note on (non-)transitivity of `skos:related`:** The reader should be aware that in the SKOS data model `skos:related` is not defined as a transitive property. A transitive `skos:related` could have unwanted consequences, as in the following example:
>
> ```
> ex:renaissance skos:related ex:humanism.
> ex:humanism skos:related ex:philosophicalAnthropology.
> ex:philosophicalAnthropology skos:related ex:philosophyOfMind.
> ex:philosophyOfMind skos:related ex:cognitiveScience.
> ```
>
> Should `skos:related` be transitive, `ex:renaissance` would be then *directly* related to `ex:cognitiveScence`. While every individual statement makes sense, the inferred statement may not fit what the designer of the KOS originally intended.

Note on mixing hierarchy with association: The transitive closure of `skos:broader` is disjoint from `skos:related`. If resources A and B are related via `skos:related`, there must not be a chain of `skos:broader` relationships from A to B. The same holds of `skos:narrower`.

2.4 Documentary Notes

Semantic relationships are crucial to the definition of concepts, as many KOS guidelines emphasize it. However, next to these structured characterizations, concepts sometimes have to be further defined using human-readable ("informal") documentation, such as *scope notes* or *definitions*.

SKOS provides a `skos:note` property for general documentation purposes. Inspired by existing KOS guidelines, such as [*ISO2788*] or [*BS8723-2*], this property is further specialized into `skos:scopeNote`, `skos:definition`, `skos:example`, and `skos:historyNote` to fit more-specific types of documentation.

`skos:scopeNote` supplies some, possibly partial, information about the intended meaning of a concept, especially as an indication of how the use of a concept is limited in indexing practice. The following example is adapted from [*ISO2788*]:

```
ex:microwaveFrequencies skos:scopeNote
    "Used for frequencies between 1GHz to 300Ghz"@en.
```

`skos:definition` supplies a complete explanation of the intended meaning of a concept. The following example is adapted from [*ISO2788*]:

```
ex:documentation skos:definition
    "the process of storing and retrieving information
    in all fields of knowledge"@en.
```

`skos:example` supplies an example of the use of a concept:

```
ex:organizationsOfScienceAndCulture skos:example
    "academies of science, general museums, world fairs"@en.
```

`skos:historyNote` describes significant changes to the meaning or the form of a concept:

```
ex:childAbuse skos:historyNote
    "estab. 1975; heading was: Cruelty to children [1952-1975]"@en.
```

In addition to these notes that are intended for users of a concept scheme, SKOS includes two specializations of `skos:note` that are useful for KOS managers or editors: `skos:editorialNote` and `skos:changeNote`.

skos:editorialNote supplies information that is an aid to administrative housekeeping, such as reminders of editorial work still to be done, or warnings in the event that future editorial changes might be made:

```
ex:doubleclick skos:editorialNote "Review this term after company mer
                                  complete"@en.
ex:folksonomy skos:editorialNote "Check spelling with Thomas Vander W
```

skos:changeNote documents fine-grained changes to a concept, for the purposes of administration and maintenance:

```
ex:tomato skos:changeNote
  "Moved from under 'fruits' to under 'vegetables' by Horace Gray"@en
```

It is important to notice that the hierarchical link between `skos:note` and its different specializations allows all the documentation associated with a concept to be retrieved in a straightforward way. Every `skos:definition` is a `skos:note`, every `skos:scopeNote` is a `skos:note`, and so on.

As illustrated above, SKOS documentation properties can be simply used with RDF plain literals. Section 4.2 will show that there are other possible patterns, as the range of these properties is not be restricted to literals. One important feature of simple literals, however, is the ability to use language tags, as done for labeling properties. Documentation may thus be provided in multiple languages:

```
ex:pineapples rdf:type skos:Concept;
  skos:prefLabel "pineapples"@en;
  skos:prefLabel "ananas"@fr;
  skos:definition "The fruit of plants of the family Bromeliaceae"@en
  skos:definition
        "Le fruit d'une plante herbacée de la famille des broméliacée
```

Before concluding this section, it is important to note that other, non-SKOS properties could be used to document concepts. The dct:creator property from Dublin Core [*DC*] can for instance be used to point to a person that created the concept:

```
ex:madagascarFishEagle dct:creator [ foaf:name "John Smith" ].
```

2.5 Concept Schemes

Concepts can be created and used as stand-alone entities. However, especially in indexing practice, concepts usually come in carefully compiled vocabularies, such as thesauri or classification schemes. SKOS offers the means of representing such KOSs using the skos:ConceptScheme class.

The following example shows how to define a concept scheme resource (representing a thesaurus) and to describe that resource using the dct:title and dct:creator properties from Dublin Core [*DC*]:

```
ex:animalThesaurus rdf:type skos:ConceptScheme;
  dct:title "Simple animal thesaurus";
  dct:creator ex:antoineIsaac.
```

Once the concept scheme resource has been created, it can be linked with the concepts it contains using the `skos:inScheme` property:

```
ex:mammals rdf:type skos:Concept;
  skos:inScheme ex:animalThesaurus.
ex:cows rdf:type skos:Concept;
  skos:broader ex:mammals;
  skos:inScheme ex:animalThesaurus.
ex:fish rdf:type skos:Concept;
  skos:inScheme ex:animalThesaurus.
```

In order to provide an efficient access to the entry points of broader/narrower concept hierarchies, SKOS defines a `skos:hasTopConcept` property. This property allows one to link a concept scheme to the (possibly many) most general concepts it contains, as in the (continued) animal thesaurus example:

```
ex:animalThesaurus rdf:type skos:ConceptScheme;
  skos:hasTopConcept ex:mammals;
  skos:hasTopConcept ex:fish.
```

Concept schemes are designed to represent traditional vocabularies, and designers are encouraged to follow existing KOS guidelines (e.g., [*ISO2788*] or [*BS8723-2*]) when compiling a SKOS concept scheme. For example, as described in Section 2.2, it is recommended that no two concepts have the same preferred lexical label in a given language when they belong to the same concept scheme.

The reader should however be aware that there are some subtle differences between SKOS concept schemes and "traditional" KOSs, mainly due to the Semantic Web context for SKOS. Section 4.6 of the *SKOS Reference* [*SKOS-REFERENCE*] gives an account of these differences. One important feature of SKOS is that it is possible for the same concept to be linked to several concept schemes, using the `skos:inScheme` property. This will be discussed in the next section.

Finally, it is important to notice that the SKOS vocabulary only offers limited support for containment of KOS information in a concept scheme. `skos:inScheme` and `skos:hasTopConcept` link concept schemes and concepts. Yet, there is no mechanism in SKOS to record that a specific statement concerning these concepts, e.g. a `skos:broader` assertion, pertains to a specific concept scheme, whereas a KOS is usually seen as consisting of both its concepts and the links that define them. The interested reader is referred to Section 5.3 for a discussion on this topic.

---

3 Networking Knowledge Organization Systems on the

Semantic Web

Representing a KOS with SKOS not only serves as a publication mechanism, but also allows it to participate in a network of concept schemes. On the Semantic Web the true potential of data is unleashed when it is interlinked. As concepts from different concept schemes are connected together they begin to form a distributed, heterogeneous global concept scheme. A web of concept schemes can serve as the foundation for new applications that allow meaningful navigation between KOSs. This section introduces the SKOS features that enable the interlinking of concept schemes and explains how to relate conceptual resources to other resources on the Semantic Web.

3.1 Mapping Concept Schemes

Every SKOS concept is assigned a URI [*COOLURIS*], which makes it possible to unambiguously reference a concept in any SKOS application. This can be especially useful for establishing semantic relations between pre-existing concepts. Such *mappings* are crucial for applications such as information retrieval tools that use several KOSs at the same time, where these KOSs have overlapping scopes and need to be semantically reconciled; examples can be found in the *SKOS Use cases and Requirements* document [*SKOS-UCR*].

A crucial feature of mapping is the possibility to state that two concepts from different schemes have comparable meanings, and to specify how these meanings compare, even though they come from different contexts and possibly follow different modeling principles [*BS8723-4*]. Conceptual mappings are expected to be a key advantage of making KOSs available on the Semantic Web using SKOS.

SKOS provides several properties that map concepts between different concept schemes. This can be done by asserting that two concepts have a similar meaning, using the `skos:exactMatch` and `skos:closeMatch` properties. Two concepts from different concept schemes can also be mapped using properties that parallel the semantic relations introduced in Section 2.3: `skos:broadMatch`, `skos:narrowMatch` and `skos:relatedMatch`.

Consider the following example, where two concept schemes represent different views on animals:

```
ex1:referenceAnimalScheme rdf:type skos:ConceptScheme;
   dct:title "Extensive list of animals"@en.
ex1:animal rdf:type skos:Concept;
   skos:prefLabel "animal"@en;
   skos:inScheme ex1:referenceAnimalScheme.
ex1:platypus rdf:type skos:Concept;
   skos:prefLabel "platypus"@en;
   skos:inScheme ex1:referenceAnimalScheme.

ex2:eggSellerScheme rdf:type skos:ConceptScheme;
```

```
   dct:title "Obsessed egg-seller's vocabulary"@en.
ex2:eggLayingAnimals rdf:type skos:Concept;
   skos:prefLabel "animals that lay eggs"@en;
   skos:inScheme ex2:eggSellerScheme.
ex2:animals rdf:type skos:Concept;
   skos:prefLabel "animals"@en;
   skos:inScheme ex2:eggSellerScheme.
ex2:eggs rdf:type skos:Concept;
   skos:prefLabel "eggs"@en;
   skos:inScheme ex2:eggSellerScheme.
```

It is possible to map the concepts in `ex1:referenceAnimalScheme` to the concepts in `ex2:eggSellerScheme` by using the mapping assertions below:

```
ex1:platypus skos:broadMatch ex2:eggLayingAnimals.
ex1:platypus skos:relatedMatch ex2:eggs.
ex1:animal skos:exactMatch ex2:animals.
```

A `skos:closeMatch` assertion indicates that two concepts are sufficiently similar that they can be used interchangeably in applications that consider the two concept schemes they belong to. However, `skos:closeMatch` is not defined as transitive, which prevents such similarity assessments to propagate beyond these two schemes. If a concept `ex1:A` is a close match for another concept `ex2:B` which is itself a close match for `ex3:C`, it **does not** follow from the SKOS data model that `ex1:A` is a close match for `ex3:C`.

`skos:exactMatch` also indicates semantic similarity—it is a sub-property of `skos:closeMatch`. However, it denotes an even higher degree of closeness: the two concepts have equivalent meaning, and the link can be exploited across a wider range of applications and schemes. `skos:exactMatch` is indeed transitive: if a concept `ex1:A` is an exact match for another concept `ex2:B` which is itself an exact match for `ex3:C`, it **does** follow from the SKOS data model that `ex1:A` is an exact match for `ex3:C`.

> **Note on `skos:exactMatch` vs. `owl:sameAs`:** SKOS provides `skos:exactMatch` to map concepts with equivalent meaning, and intentionally does not use `owl:sameAs` from the OWL ontology language [*OWL*]. When two resources are linked with `owl:sameAs` they are considered to be the same resource, and triples involving these resources are merged. This does not fit what is needed in most SKOS applications. In the above example, `ex1:animal` is said to be equivalent to `ex2:animals`. If this equivalence relation were represented using `owl:sameAs`, the following statements would hold for `ex:animal`:
>
> ```
> ex1:animal rdf:type skos:Concept;
>    skos:prefLabel "animal"@en;
>    skos:inScheme ex1:referenceAnimalScheme.
>    skos:prefLabel "animals"@en;
>    skos:inScheme ex2:eggSellerScheme.
> ```
>
> This would make `ex:animal` inconsistent, as a concept cannot

possess two different preferred labels in the same language. Had the concepts been assigned other information, such as semantic relationships to other concepts, or notes, these would be merged as well, causing these concepts to acquire new meanings.

By convention, mapping properties are used to represent links that have the same intended meaning as the "standard" semantic properties, but with a different application scope. One might say that mapping relationships are less *inherent* to the meaning of the concepts they involve. From the point of view of the original designer of a mapped KOS, they might even sometimes be wrong.

Mapping properties are expected to be useful in *specific* applications that use multiple, conceptually overlapping KOSs. By convention, mapping relationships are expected to be asserted between concepts that belong to different concept schemes.

The reader should be aware that according to the SKOS data model, the mapping properties that "mirror" a given semantic relation property are also sub-properties of it in the RDFS sense. For instance, `skos:broadMatch` is a sub-property of `skos:broader`. Consequently, every assertion of `skos:broadMatch` between two concepts leads by inference to asserting a `skos:broader` between these concepts.

3.2 Re-using and Extending Concept Schemes

Linking concepts by means of mappings is not the only way to interlink concept schemes. The use of URIs on the Semantic Web allows resources to be shared and reused in a distributed fashion. As a result it is possible for a SKOS concept to participate in several concept schemes at the same time. For example, a SKOS publisher can choose to locally extend an existing concept scheme by declaring any new concepts that may be needed and simply *linking* to concepts that have already been defined in the existing scheme.

Extension of a KOS can be especially useful when its designers (or third-party KOS publishers) want to achieve a better coverage of a domain or sub-domain, while following the principles that guided the design of the existing KOS—e.g., by re-using some of its concepts. Explicit KOS extension and re-use can also be used as a modularization mechanism, when a family of articulated KOSs (for instance microthesauri that belong to an overarching vocabulary) is designed to cover several domains and its designers want to allow specific applications to operate on given subsets of concepts.

A new concept scheme can re-use existing concepts using the `skos:inScheme` property. Consider the example below, where a first concept scheme for animals defines a concept for "cats":

```
ex1:referenceAnimalScheme rdf:type skos:ConceptScheme;
   dct:title "Reference list of animals"@en.
```

```
ex1:cats rdf:type skos:Concept;
   skos:prefLabel "cats"@en;
   skos:inScheme ex1:referenceAnimalScheme.
```

The creator of another concept scheme devoted to cat descriptions can freely include the reference `ex1:cats` concept in her scheme, and then reference it as follows:

```
ex2:catScheme rdf:type skos:ConceptScheme;
   dct:title "The Complete Cat Thesaurus"@en.

ex1:cats skos:inScheme ex2:catScheme.

ex2:abyssinian rdf:type skos:Concept;
   skos:prefLabel "Abyssinian Cats"@en;
   skos:broader ex1:cats;
   skos:inScheme ex2:catScheme.

ex2:siamese rdf:type skos:Concept;
   skos:prefLabel "Siamese Cats"@en;
   skos:broader ex1:cats;
   skos:inScheme ex2:catScheme.
```

Note that the information source defining the new concept scheme does not replicate information about the `ex1:cats` concept, such as its preferred label. Assuming `ex1:cats` has been published, a Semantic Web application is able to retrieve the information for this concept by simply resolving the concept's URI (`http://www.example.com/1/cats`).

> **Note on `owl:imports` and re-using KOSs**: The `owl:imports` property provides a mechanism for importing the assertions of one OWL ontology into another. `owl:imports` may be used with SKOS vocabularies to provide a special case of re-use/extension where a concept scheme "imports" another concept scheme as a whole. To continue the example above, this is achieved by including the following statement in the source defining `ex2:catScheme`:
>
> ```
> ex2:catScheme owl:imports ex1:referenceAnimalScheme.
> ```
>
> Using `owl:imports` in this way has some ramifications. First, the domain and range of `owl:imports` is `owl:Ontology`, while `skos:ConceptScheme` is defined as an `owl:Class`. Thus asserting that a concept scheme imports another via `owl:imports` leads to the consequence that the instances of `skos:conceptScheme` involved in the import are also inferred to be instances of `owl:Ontology`. This in turn results in an OWL Full ontology (due to the dual use of a URI as a class and ontology, see Section 4.2 of the *OWL Semantics* document [*OWL-SEMANTICS*]).
>
> Second, under the OWL Full semantics (see Section 5.3 of the *OWL Semantics* [*OWL-SEMANTICS*]), the intended interpretation of `owl:imports` is that the RDF graph retrieved from the imported URI is added to the importing graph. Users should be aware of this, and any

alternative interpretations should be avoided. In particular, there is no logical dependency between `skos:inScheme` and `owl:imports`: the use of `owl:imports` will not result in the presence of any `skos:inScheme` statements other than the ones already asserted in the imported graph. If we consider the example above, `owl:imports` has been used to state that one concept scheme logically imports another. But even though `ex1:referenceAnimalScheme` contains the triple

```
ex1:Elephant skos:inScheme ex1:referenceAnimalsScheme.
```

the triple

```
ex1:Elephant skos:inScheme ex2:catScheme.
```

should **not** be inferred to be present in the graph defining `ex2:catScheme`.

If an application is concerned with practical provenance or ownership information, additional steps may be required in order to maintain the provenance or assert the authority of imported triples, as mentioned in Section 5.3.

3.3 Subject Indexing and SKOS

Though formally not belonging to the features defining a KOS, the link between a concept and the resources which are about this concept is fundamental in many KOS applications, such as document indexing and document retrieval. This becomes even more important in a Semantic Web context, where there is a crucial need to annotate documents with conceptual units which define their subject.

While the SKOS vocabulary itself does not include a mechanism for associating an arbitrary resource with a `skos:Concept`, implementors can turn to other vocabularies. Dublin Core, for instance, provides a `dct:subject` property [*DC*]:

```
ex1:platypus rdf:type skos:Concept;
   skos:prefLabel "platypus"@en.

<http://en.wikipedia.org/wiki/Platypus> rdf:type foaf:Document;
   dct:subject ex1:platypus.
```

Note that a single resource can have several subjects, and hence be involved in several `dct:subject` statements. These subjects can clearly come from different concept schemes, resulting for instance from a distributed annotation process.

---

4 Advanced SKOS: When KOSs are not Simple

Anymore

Beyond the above mentioned features, SKOS proposes a number of vocabulary elements or guidelines that deal with more-advanced representation needs, making SKOS compatible with a broad range of KOS modeling approaches. These are especially designed to meet requirements which were raised in the *SKOS Use Cases and Requirements* [*SKOS-UCR*], but which were only present in a smaller number of use cases:

- Grouping of concepts based on specific criteria,
- Advanced documentation by means of complex resources,
- Establishing relationships between labels of concepts,
- Creation of complex concepts from simple ones (coordination),
- Accessing transitive hierarchical relationships,
- Representing notations for concepts.

This section concludes with a general note on the extensibility of the SKOS model, paving the way for even more specialized refinements of the vocabulary presented in this Primer.

4.1 Collections of Concepts

SKOS makes it possible to define meaningful groupings or "collections" of concepts. Such groupings are normally rendered in thesauri as in the following example:

```
milk
  <milk by source animal>
    cow milk
    goat milk
    buffalo milk
```

These collections can be used to represent "arrays" in thesaurus terminology, in which the term "milk by source animal" is a "node label" [*WillpowerGlossary*]. There is consensus that a node label does **not** represent a label for a concept in its own right. Therefore, specific entities have to be introduced to represent them.

Labeled Collections

To correctly model such concept collection structures, SKOS introduces a `skos:Collection` class. Instances of this class group specific concepts by means of the `skos:member` property, as in the following example:

```
ex:milk rdf:type skos:Concept;
  skos:prefLabel "milk"@en.
ex:cowMilk rdf:type skos:Concept;
  skos:prefLabel "cow milk"@en;
  skos:broader ex:milk.
```

```
ex:goatMilk rdf:type skos:Concept;
  skos:prefLabel "goat milk"@en;
  skos:broader ex:milk.
ex:buffaloMilk rdf:type skos:Concept;
  skos:prefLabel "buffalo milk"@en;
  skos:broader ex:milk.

_:b0 rdf:type skos:Collection;
   skos:prefLabel "milk by source animal"@en;
   skos:member ex:cowMilk;
   skos:member ex:goatMilk;
   skos:member ex:buffaloMilk.
```

Note that in the example above the collection was defined as a blank node, i.e. no defined URI was allocated. URIs may be allocated to collections, but usually this is not necessary. Also, `skos:prefLabel` has been used to assign a lexical label to the Collection, as this property (as other SKOS labeling properties) can be used with non-conceptual resources.

Ordered Collections

Sometimes it is important to capture the order of concepts in a collection, such as when concepts are listed in alphabetical or chronological order. To define an ordered collection of concepts the `skos:OrderedCollection` class is used, together with the `skos:memberList` property. This property links an instance of `skos:OrderedCollection` to a (possibly blank) node of type `rdf:List`, following the pattern that enables the definition of RDF collections [*RDF-PRIMER*]. For example:

```
ex:infants rdf:type skos:Concept;
  skos:prefLabel "infants"@en.
ex:children rdf:type skos:Concept;
  skos:prefLabel "children"@en.
ex:adults rdf:type skos:Concept;
  skos:prefLabel "adults"@en.

_:b0 rdf:type skos:OrderedCollection;
   skos:prefLabel "people by age"@en;
   skos:memberList _:b1.
_:b1 rdf:first ex:infants;
   rdf:rest _:b2.
_:b2 rdf:first ex:children;
   rdf:rest _:b3.
_:b3 rdf:first ex:adults;
   rdf:rest rdf:nil.
```

SKOS Collections, Semantic Relations and Systematic Displays

Note that, according to the SKOS data model, collections are disjoint from concepts. It is therefore impossible to use SKOS semantic relations (see Section 2.3) to have a collection directly fit into a SKOS semantic network. In other words, grouping concepts into collections does not replace assertions about the concepts' place in a concept scheme. For instance, in the above "milk" example, all source-defined milks must be explicitly attached to a more generic `ex:milk` using the `skos:broader` property:

```
ex:cowMilk skos:broader ex:milk.
ex:goatMilk skos:broader ex:milk.
ex:buffaloMilk skos:broader ex:milk.
```

A systematic (hierarchical) display can then be generated including the concept grouping "milk by source animal", as presented in the example introducing this sub-section. The `skos:broader` hierarchy and the collection membership information can be used for this, but the process still requires a dedicated algorithm, the implementation of which is left to specific applications.

One may wonder whether using collections is desirable, as they add complexity to the representations applications have to manipulate. In fact, for some cases, e.g. when KOSs are mainly intended as navigation hierarchies, it seems more intuitive to represent "node labels" or "guide terms" as instances of `skos:Concept`, and to use normal semantic relationships for linking them to other concepts. Take the following variant of the "milk" example:

```
ex3:milkBySourceAnimal rdf:type skos:Concept;
   skos:prefLabel "milk by source animal"@en;
   skos:broader ex3:milk;
   skos:narrower ex3:cowMilk;
   skos:narrower ex3:goatMilk;
   skos:narrower ex3:buffaloMilk.
```

The choice between the two representation options remains open, depending on the application at hand. Readers should however be aware that not using collections, even if that is more intuitive, may result in a harmful loss of semantic accuracy. For many description applications, for instance, "node labels" are entities of a really specific nature, and must not be used as object indices alongside "normal" concepts. Representing them as mere concepts is therefore clearly not a best practice.

4.2 Advanced Documentation Features

As shown in Section 2.4, SKOS allows concepts to be annotated by attaching various notes to them. It is worth noticing that the SKOS Reference does not restrict the range of resources that assertions can use in the object position. This leads to different usage patterns, three of which are explained—and recommended—in this document.

Documentation as an RDF literal

Here, documentation statements have simple RDF literals as objects, as illustrated by *all* examples of Section 2.4. This is the simplest way to document concepts, and it is expected to fit most common applications.

Documentation as a Related Resource Description

In this second pattern, the object of a documentation statement consists of a

general non-literal RDF node—that is, a resource node (possibly blank) that can be the subject of further RDF statements [*RDF-PRIMER*]. This is especially useful to represent with RDF more information about the documentation itself, such as its creator or creation date. This is typically done using the RDF rdf:value utility property, as in the following example, which uses a blank node:

```
ex:tomato skos:changeNote [
  rdf:value "Moved from under 'fruits' to under 'vegetables'"@en;
  dct:creator ex:HoraceGray;
  dct:date "1999-01-23"
].
ex:HoraceGray rdf:type foaf:Person; foaf:name "Horace Gray".
```

Documentation as a Document Reference

A third option consists of introducing, as the object of a documentation statement, the URI of a *document*, for instance a Web page. Note that this pattern, closely related to the previous one, also allows the definition of further metadata for this document using RDF:

```
ex:zoology skos:definition ex:zoology.txt.
ex:zoology.txt dct:creator ex:JohnSmith.
```

4.3 Relationships between Labels

Some applications require the creation of explicit links between the labels associated with concepts. For example, consider the relationship between a preferred label for a concept "Corporation" and its abbreviation "Corp." coined as an alternative label, or a translation link between two labels in different languages: `"Cow"@en` and `"Vache"@fr`. The use of SKOS lexical labeling properties, e.g. `skos:prefLabel`, is restricted to RDF literals. Therefore these labels cannot be the subject of an RDF statement, and a direct relationship cannot be asserted between them.

To solve this representation issue, the SKOS vocabulary has been augmented with an optional extension for labels, SKOS-XL [*SKOS-REFERENCE*]. This extension introduces a `skosxl:Label` class that allows labels to be treated as first-order RDF resources. Each instance of this class shall first be attached to a single RDF literal via the `skosxl:literalForm` property. Consider the example where the concept "Food and Agriculture Organization" is labeled by both the official name and the acronym of the institution. The two labels can be represented in the following way:

```
ex:FAOlabel1 rdf:type skosxl:Label;
  skosxl:literalForm "Food and Agriculture Organization"@en.
ex:FAOlabel2 rdf:type skosxl:Label;
  skosxl:literalForm "FAO"@en.
```

`skosxl:Label` instances can then be related to concepts using properties (`skosxl:prefLabel`, `skosxl:altLabel`, `skosxl:hiddenLabel`) that mirror the

standard literal-based labeling constructs. Finally, these instances can be linked together by `skosxl:labelRelation` statements:

```
ex:FAO rdf:type skos:Concept;
  skosxl:prefLabel ex:FAOlabel1;
  skosxl:altLabel ex:FAOlabel2.
ex:FAOlabel2 skosxl:labelRelation ex:FAOlabel1.
```

Such a solution is however not complete: an "acronym-sensitive" application would miss the actual information that the two labels are indeed in an *acronymy* relationship. Such an application would also miss the *direction* of the link. SKOS-XL users are therefore encouraged to specialize `skosxl:labelRelation` so as to fit their application-specific requirements, as in the following:

```
ex:isAcronymOf rdfs:subPropertyOf skosxl:labelRelation.
ex:FAOlabel2 ex:isAcronymOf ex:FAOlabel1.
```

Note that the SKOS-XL data model ensures that using such a pattern remains compatible with the standard SKOS labeling practice. If an instance of `skosxl:Label` is attached to a concept by, say, a `skosxl:altLabel` statement, it follows from the SKOS-XL data model that the literal form of the `skosxl:Label` instance is related to this concept by a standard `skos:altLabel` statement. In the above example, `ex:FAO` therefore has `"FAO"@en` as alternative (literal) label.

4.4 Coordinating Concepts

Indexing practices involving thesauri and other KOSs often include the notion of *coordination*. Coordination is an activity in which concepts from a KOS are combined. In general there are two kinds of coordination: pre-coordination and post-coordination [*WillpowerGlossary*]. The key distinction between the two hinges on when the actual coordination occurs in relation to an information retrieval event.

Pre-coordination is done prior to information retrieval, by a KOS maintainer, or by an indexer who is using a KOS—for example, if an indexer takes two existing concepts from a concept scheme, such as "Bicycles" and "Repairing", and explicitly combines them with a given syntax such as "Bicycles--Repairing" to index a particular document.

Post-coordination on the other hand is performed as part of an information retrieval task—for example, if a given document is indexed with two distinct concepts "Bicycles" and "Repairing" and a user decides to perform a search for all documents that are indexed with "Bicycles" and "Repairing".

Post-coordination as an information retrieval activity lends itself to *indirect* representation as a SPARQL query to access RDF data [*SPARQL*]. For example, given two distinct concepts:

```
ex:bicycles skos:prefLabel "Bicycles"@en.
```

```
ex:repairing skos:prefLabel "Repairing"@en.
```

one could construct a SPARQL query to return only the documents that are indexed with both concepts

```
SELECT ?document
WHERE {
  ?document dct:subject ex:bicycles.
  ?document dct:subject ex:repairing.
}
```

However the SKOS vocabulary itself does not provide any mechanism for expressing that a given concept consists of a pre-coordination of other concepts. Of course it is perfectly feasible to extend SKOS to establish a pattern for representing coordinated concepts. For example it has been suggested that a new property such as `ex:coordinationOf` could be established:

```
ex:coordinationOf a rdf:Property;
  rdfs:domain skos:Concept;
  rdfs:range rdf:List.
```

which could then be used in assertions such as:

```
ex:bicyclesRepairing a skos:Concept;
  ex:coordinationOf (ex:bicycles ex:repairing);
  skos:prefLabel "Bicycles--Repairing"@en.
```

It has also been suggested that OWL itself could be used to coordinate concepts:

```
ex:bicyclesRepairing a skos:Concept;
  owl:intersectionOf (ex:bicycles ex:repairing);
  skos:prefLabel "Bicycles--Repairing"@en.
```

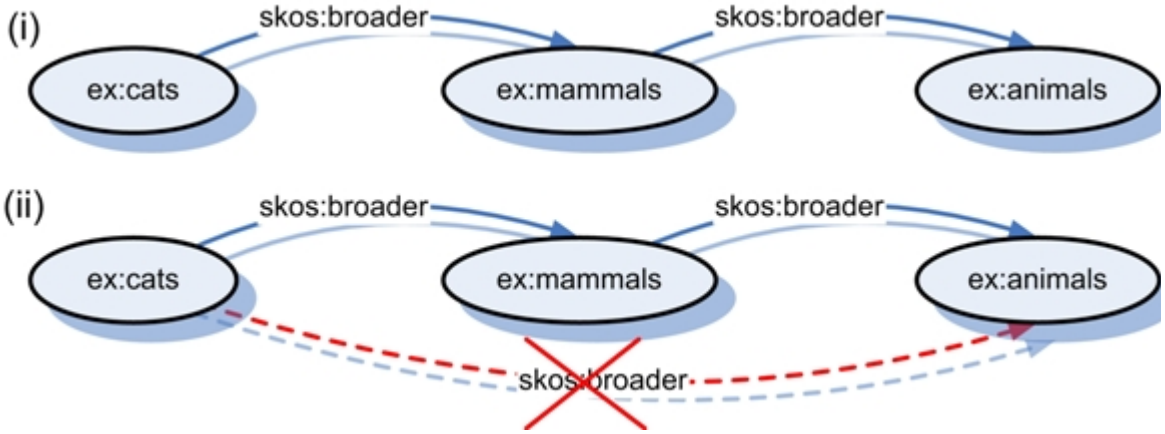
However, established patterns for pre-coordinations of this kind have not yet emerged in the SKOS community. `ex:coordinationOf` (or some equivalent extension), and the ramifications of using SKOS with OWL have not been explored fully enough yet to warrant inclusion in the SKOS vocabulary. Rather than commit to a design pattern that has not been proven useful, the Semantic Web Deployment Group decided to postpone the issue of coordination, to allow extension patterns to organically emerge as SKOS is deployed. The hope is that as successful patterns are established, they can be published on the Web as an extension vocabulary to SKOS and documented as a W3C Note or some equivalent.

4.5 Transitive Hierarchies

As described in Section 2.3.1, the properties used to represent KOS hierarchies, `skos:broader` and `skos:narrower`, are not defined as transitive. As

shown in Fig. 4.5.1 (i) & (ii), this means that their semantics do **not** support inferences of the type: *if "animals" is broader than "mammals" and "mammals" is broader than "cats", then "animals" is broader than "cats"*.

Figure 4.5.1: `skos:broader` is not transitive



Dotted arrows represent statements inferred from the SKOS data model. Solid arrows represent asserted statements.

For the applications that require such semantics—for instance to perform query expansion—SKOS features two specific properties, `skos:broaderTransitive` and `skos:narrowerTransitive`. These are defined as transitive super-properties of `skos:broader` and `skos:narrower` [*SKOS-REFERENCE*]. This pattern enables, using a Semantic Web inferencing tool, access to the "transitive closure" of a hierarchy expressed with `skos:broader` and `skos:narrower`.

Consider the example of Fig. 4.5.1 (i):

```
ex:animals skos:prefLabel "animals"@en.
ex:mammals skos:prefLabel "mammals"@en;
  skos:broader ex:animals.
ex:cats skos:prefLabel "cats"@en;
  skos:broader ex:mammals.
```

When reading the above triples, a reasoner makes use the definition of `skos:broaderTransitive` as a super-property of `skos:broader` to infer the following statements:

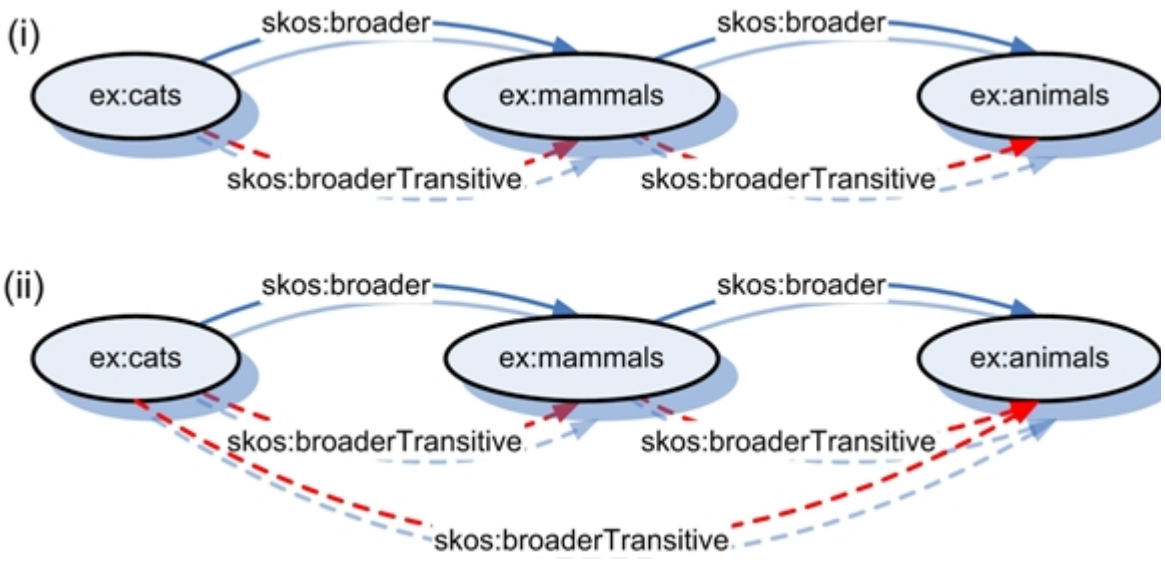
```
ex:cats skos:broaderTransitive ex:mammals.
ex:mammals skos:broaderTransitive ex:animals.
```

The transitivity of `skos:broaderTransitive` then causes the desired statement to be inferred:

```
ex:cats skos:broaderTransitive ex:animals.
```

These two steps are showed in the following figure:

Figure 4.5.2: inferring a transitive hierarchy from asserted `skos:broader` statements



Dotted arrows represent statements inferred from the SKOS data model. Solid arrows represent asserted statements.

The use of the `skos:broaderTransitive` super-property allows communities of practice to exploit transitive interpretations of hierarchical networks as they see fit, while not interfering with the semantics of `skos:broader`, which cannot enforce such transitivity. Intuitively, one can interpret `skos:broader` statements as explicitly asserted *direct parent* links, while `skos:broaderTransitive` is used to reflect more-general (and possibly indirect) *ancestor* relationships.

> **Note on supposed "transitiveness inheritance":** the super-property link between `skos:broader` and `skos:broaderTransitive` may look counter-intuitive at first glance. Here, a non-transitive property is defined as a child of a transitive one, while not inheriting its transitiveness. This is however fully compliant with RDFS/OWL semantics for `rdfs:subPropertyOf` [*OWL*]: a property P is a sub-property of Q if and only if every time P holds between two resources, then Q also holds between them. This does not enforce any transitiveness inheritance: on the contrary, the set of all couples of resources related by P (its *graph*), as a subset of Q's, is likely to miss some of the couples that make Q transitive.

4.6 Notations

Some KOSs, for example classification systems such as the Universal Decimal Classification [*UDC*], use *notations* (or *captions*) as the primary means of access to the concepts they contain. Notations are symbols which are not normally recognizable as words or sequences of words in any natural language

and are thus usable independently of natural-language contexts. They are typically composed of digits, complemented with punctuation signs and other characters, as in the following UDC example:

```
512 Algebra
512.6 Special branches of algebra
```

SKOS allows notations to be represented in two ways, depending on the priorities of the concept scheme publisher. The first, preferred technique is to use the `skos:notation` property. This property allows a concept to be attached to an RDF typed literal—a literal with an explicit datatype [*RDF-PRIMER*]. The datatype of the literal specifies a syntax encoding scheme, which fits the usage of notations in the concerned KOS. The value of the literal is the notation itself (in this case the classification code itself):

```
ex:udc512 skos:prefLabel "Algebra"@en ;
  skos:notation "512"^^ex:UDCNotation .
```

Section 6.5.1 of the *SKOS Reference* gives more detail on how to handle datatypes [*SKOS-REFERENCE*]. This approach can be especially useful if a KOS publisher wants to provide users with processing rules that are specific to the KOS's notation scheme. For instance, many classification systems have specific syntax rules which allow complex notations to be decomposed, leading to the linking of the corresponding concept to other, simpler concepts. Also, this pattern can help creators of SKOS tools and KOS publishers who want to have notations displayed in a dedicated way.

However, the management of such datatypes can be cumbersome. Further, the previous pattern is not really needed when publishers consider the notations themselves to be simple language-independent labels. In such cases, it is possible to use one SKOS labeling property, for instance `skos:prefLabel`, without any language tag, as in:

```
ex:udc512 skos:prefLabel "Algebra"@en ;
  skos:notation "512"^^ex:UDCNotation ;
  skos:prefLabel "512" .
```

Note that it is unlikely that notations represented in such a manner will benefit from notation-specific mechanisms (such as display procedures) in SKOS tools. By default, users should expect these notations to be treated, in accordance with the SKOS model, as mere labels.

4.7 On Specializing the SKOS Model

SKOS is intended to serve as a common denominator between different modeling approaches. As such, the current vocabulary specification will allow many existing KOSs to be ported to the Semantic Web. However, the great variety of KOS models makes it impossible to capture every detail of these models while still retaining the first "S" ("simple") in "SKOS".

Applications that require finer granularity will greatly benefit from SKOS's being a Semantic Web vocabulary. SKOS can indeed be seamlessly *extended* to suit the specific needs of a particular KOS community while retaining compatibility with applications that are based on the core SKOS features.

This can mostly be done by *specializing* existing SKOS constructs into more-specific ones. Users can create their own properties and classes and attach them to the standard SKOS vocabulary elements by using the `rdfs:subPropertyOf` and `rdfs:subClassOf` properties from the RDF Schema vocabulary [*RDF-PRIMER*].

The example in Section 4.3 illustrates how `skosxl:labelRelation` can be specialized into a semantically richer property devoted to acronym link representation. Other uses are possible, such as creating different "flavors" of the properties `skos:broader` and `skos:narrower`. Thesaurus standards indeed identify a small number of kinds of hierarchical relation, such as generic, part-whole, or instance-class [*ISO2788*]. The SKOS approach allows an application designer to create new properties to capture this distinction, and to declare them as sub-properties of `skos:broader`:

```
ex:broaderGeneric rdfs:subPropertyOf skos:broader.
ex:broaderPartitive rdfs:subPropertyOf skos:broader.
ex:broaderInstantive rdfs:subPropertyOf skos:broader.
```

Every `ex:broaderPartitive` statement between two concepts, for instance, can be formally interpreted by a proper Semantic Web reasoning engine. This interpretation will yield the inference of a `skos:broader` statement between these concepts—a piece of information which may then be exploited by basic SKOS tools.

> **Note on tampering with the SKOS vocabulary itself:** In general, it is best to avoid stating triples where a URI from the SKOS vocabulary is in the *subject* position. By doing so, one may alter the SKOS data model and introduce unwanted side effects. This may then compromise the interoperability of vocabularies. If one wants to adapt the behavior of the "built-in" vocabulary to specific cases, one should first consider introducing one's own constructs as sub-classes or sub-properties.

Of course the creators of extensions to SKOS are encouraged to publish them, e.g., using the SKOS public mailing list (public-esw-thes@w3.org). Such extensions might correspond to shared concerns and thus be re-usable across different applications. In turn, re-use is likely to bring community feedback, helping to enhance the quality of published extensions.

---

5 Combining SKOS with other Modeling Approaches

As seen above, SKOS is an RDF/OWL vocabulary which can be seamlessly

extended to fit specific requirements. Likewise, SKOS features can also be used on the Semantic Web as a complement to other modeling vocabularies. This section gives examples of re-using SKOS labeling properties to describe resources that are not necessarily SKOS concepts. It then deals with the specific problem of articulating SKOS concepts with *classes* as defined by the ontology language OWL.

> **Note:** this section deals with the issues arising when an application requires SKOS features to be used in coordination with other modeling approaches. Users not having such a requirement may skip it.

5.1 Use of Labels Outside of SKOS

It is possible to use SKOS labeling properties to label resources that are not of type `skos:Concept`. Consider these triples that describe Tim Berners-Lee:

```
<http://www.w3.org/People/Berners-Lee/card#i> rdf:type foaf:Person;
  foaf:name "Timothy Berners-Lee";
  rdfs:label "TBL";
  skos:prefLabel "Tim Berners-Lee"@en.
```

An application that wishes to display a label for this resource is able to identify "Tim Berners-Lee" as the preferred label instead of having to choose between the equally compatible labels `rdfs:label` "TBL" or the `foaf:name` "Timothy Berners-Lee"—these labels are compatible because `foaf:name` is a sub-property of `rdfs:label`.

Another example is human-readable labels on classes, properties and individuals in OWL ontologies, which are normally expressed using `rdfs:label` alone. Consider the following triples that describe humans:

```
ex:Human rdf:type owl:Class;
  rdfs:label "human"@en;
  rdfs:label "man"@en.
```

An application would have difficulty determining the correct label to display to the user since both labels have the same weight. The semantics of `skos:prefLabel` allow implementors to explicitly define the preferred label for a given resource. In general the ability to reuse vocabulary elements from SKOS and other RDF vocabularies as needed is what gives RDF much of its expressive power.

5.2 SKOS Concepts and OWL Classes

The SKOS Reference defines `skos:Concept` as an OWL class [*SKOS-REFERENCE*]:

```
skos:Concept rdf:type owl:Class.
```

Thus, instances of `skos:Concept` (e.g. `ex:Painting` in an art vocabulary) are in OWL terms individuals.

```
ex:Painting rdf:type skos:Concept.
```

This raises the question whether a SKOS concept instance such as `ex:Painting` can be treated as a class in its own right. For example, can users define properties of `ex:painting` such as `ex:support`:

```
ex:support rdf:type owl:DatatypeProperty.
ex:support rdfs:domain ex:Painting.
```

One might ask the question: why would someone want to do this? Well, conceptually a class such as `skos:Concept` can be seen as a metaclass: its instances are the concepts occurring in a vocabulary. So, it is conceivable that SKOS users want to specify class-level characteristics of SKOS concepts, for example that paintings have supports or that cheese has a country of origin.

It should be pointed out that SKOS does not take a stance with respect to the flavor of OWL—OWL Full or OWL-DL [*OWL-REFERENCE*]—to be used together with SKOS. OWL Full users will be able to handle the situation above by treating instances of SKOS concepts explicitly as classes, e.g. by adding statements of the form:

```
ex:Painting rdf:type owl:Class.
```

This is possible because OWL Full does not require the sets of classes and individuals to be disjoint. People who wish to use the DL flavor of OWL cannot use this metamodeling mechanism, as the disjointness condition between classes and individuals must hold for any OWL-DL ontology. The OWL-DL users interested in linking OWL classes to SKOS concepts have to keep these formally distinct. They can nevertheless use dedicated *OWL annotation properties* to bridge them, provided they can create and use their own extension for SKOS, as in:

```
ex:PaintingClass rdf:type owl:Class.
ex:PaintingConcept rdf:type skos:Concept.
ex:PaintingClass ex:correspondingConcept ex:PaintingConcept.
```

Note that at the time of writing, the recently started *OWL Working Group* [*OWL-WG*] had been chartered to handle (some forms of) metamodeling within a description-logic framework. This might allow OWL-DL users to opt for patterns that are easier to exploit.

Summarizing, the relationship between SKOS concepts and OWL classes/individuals is as follows:

- SKOS concepts are OWL individuals;
- SKOS does not take a stance on whether it must also be possible to treat SKOS concepts as OWL classes;
- The restrictions on OWL-DL prevent treating SKOS concepts as OWL

classes;
- There is an expectation that an ongoing OWL revision will alleviate the latter problem by offering some form of metamodeling.

5.3 SKOS, RDF Datasets and Information Containment

In a context of networked KOSs, some applications may require the provenance or ownership of SKOS statements to be tracked, for instance for trust purposes. A specific issue is how to establish explicit links between a concept scheme and every piece of information that is stated in the original KOS it represents, including for instance semantic relationships between concepts.

Such functionality, albeit identified as a candidate requirement [*SKOS-UCR*], is currently outside the scope of SKOS. In RDF, statements comes as context-free triples, which makes it difficult to represent containment and provenance.

However, solutions for such problems have been proposed, such as named graphs [*NAMED-GRAPHS*], and the use of RDF Datasets in SPARQL [*SPARQL*]. A SKOS concept scheme can be related to an RDF Dataset, or even asserted to be such a Dataset, which enables the creation of SPARQL queries dealing with some form of provenance or containment. Continuing the example of Section 3.2, and assuming that `ex1:referenceAnimalScheme` and `ex2:catScheme` have been managed as appropriate RDF Datasets (here, named graphs), the query

```
SELECT  ?x ?y
WHERE   {
  GRAPH ex2:catScheme { ?x skos:broader ?y }
}
```

may return `(ex2:abyssinian, ex1:cat)` as a result, while this tuple would not appear among the results of

```
SELECT  ?x ?
WHERE   {
   GRAPH ex1:referenceAnimalScheme { ?x skos:broader ?y }
}
```

Readers should nevertheless be aware that these mechanisms have not been widely used at the time of writing, and that different standard practices could emerge in the future.

References

[BS8723-2]
BS 8723-2:2005 Structured vocabularies for information retrieval. Guide.

Thesauri, British Standards Institution, London, 2005.

[BS8723-4]
BS 8723-4:2007 Structured vocabularies for information retrieval. Guide. Interoperability between vocabularies, British Standards Institution, London, 2007.

[COOLURIS]
Cool URIs for the Semantic Web, Leo Sauermann, Richard Cyganiak, Editors, W3C Interest Group Note, 3 December 2008. Latest version available at http://www.w3.org/TR/cooluris/ .

[DC]
DCMI Metadata Terms, 14 January 2008. Latest version available at http://dublincore.org/documents/dcmi-terms/ .

[ISO2788]
ISO 2788:1986 Documentation - Guidelines for the establishment and development of monolingual thesauri. Second edition. ISO TC 46/SC 9, 1986.

[ISO5964]
ISO 5964:1985 Documentation - Guidelines for the establishment and development of multilingual thesauri. First edition. ISO TC 46/SC 9, 1985.

[NAMED-GRAPHS]
Named graphs, provenance and trust, Jeremy Carroll, Christian Bizer, Patrick Hayes, Patrick Stickler, WWW 2005.

[RDF/XML-SYNTAX]
RDF/XML Syntax Specification (Revised), Dave Beckett, Editor. W3C Recommendation, 10 February 2004. Latest version available at http://www.w3.org/TR/rdf-syntax-grammar/ .

[RECIPES]
Best Practice Recipes for Publishing RDF Vocabularies. Diego Berrueta, Jon Phipps. W3C Working Draft, 23 January 2008. Latest version available at http://www.w3.org/TR/swbp-vocab-pub/ .

[OWL-WG]
OWL Working Group, http://www.w3.org/2007/OWL/.

[OWL]
OWL Web Ontology Language Reference, Mike Dean, Guus Schreiber, Editors, W3C Recommendation, 10 February 2004. Latest version available at http://www.w3.org/TR/owl-ref/ .

[OWL-SEMANTICS]
OWL Web Ontology Language Semantics and Abstract Syntax, Peter F. Patel-Schneider, Patrick Hayes, Ian Horrocks, Editors, W3C Recommendation, 10 February 2004. Latest version available at http://www.w3.org/TR/owl-semantics/ .

[RDF-PRIMER]
RDF Primer, Frank Manola, Eric Miller, Editors, W3C Recommendation, 10 February 2004. Latest version available at http://www.w3.org/TR/rdf-primer/ .

[RDF-CONCEPTS]
Resource Description Framework (RDF): Concepts and Abstract Syntax , Graham Klyne, Jeremy Carroll, Editors, W3C Recommendation, 10 February 2004. Latest version available at http://www.w3.org/TR/rdf-concepts/ .

[RFC4646]

Tags for Identifying Languages, A. Phillips , M. Davis, Editors, September 2006. Available at http://www.ietf.org/rfc/rfc4646.txt .

[SWBP-SKOS-CORE-GUIDE]
SKOS Core Guide, Alistair Miles, Dan Brickley, Editors, W3C Working Draft, 2 November 2005. Latest version available at http://www.w3.org/TR/2005/WD-swbp-skos-core-guide-20051102/ .

[SKOS-REFERENCE]
SKOS Reference, Alistair Miles, Sean Bechhofer, Editors, W3C Recommendation, 18 August 2009. Latest version available at http://www.w3.org/TR/skos-reference .

[SKOS-UCR]
SKOS Use Cases and Requirements, Antoine Isaac, Jon Phipps, Daniel Rubin, Editors, W3C Working Group Note, 18 August 2009. Latest version available at http://www.w3.org/TR/skos-ucr .

[SPARQL]
SPARQL Query Language for RDF, Eric Prud'hommeaux, Andy Seaborne, Editors, W3C Working Draft, 15 January 2008. Latest version available at http://www.w3.org/TR/rdf-sparql-query/ .

[SWBP-SKOS-CORE-SPEC]
SKOS Core Vocabulary Specification, Alistair Miles, Dan Brickley, Editors, W3C Working Draft, 2 November 2005. Available at http://www.w3.org/TR/2005/WD-swbp-skos-core-spec-20051102/ .

[SWD]
The Semantic Web Deployment Working Group, http://www.w3.org/2006/07/SWD/ .

[TURTLE]
Turtle - Terse RDF Triple Language , David Beckett, Tim Berners-Lee. W3C Team Submission, 14 January 2008. Latest version available at http://www.w3.org/TeamSubmission/turtle/ .

[UDC]
UDC - Universal Decimal Classification, UDC Consortium, http://www.udcc.org/ .

[URI]
RFC 3986 - Uniform Resource Identifiers (URI): Generic Syntax, Tim Berners-Lee, Roy Fielding, Larry Masinter, IETF, January 2005. Available at http://tools.ietf.org/html/rfc3986 .

[WillpowerGlossary]
Glossary of terms relating to thesauri and other forms of structured vocabulary for information retrieval, Stella Dextre Clarke, Alan Gilchrist, Ron Davies and Leonard Will,Willpower Information. Available at http://www.willpowerinfo.co.uk/glossary.htm .

Acknowledgments

The authors would like to thank Alistair Miles and Dan Brickley who edited the SKOS Core Guide (which this Primer is largely based on); as well as Tom Baker, Guus Schreiber and Sean Bechhofer who contributed significant portions of this text. Semantic Web Deployment Group members Tom Baker,

Margherita Sini, Quentin Reul also provided extensive reviews during the publication process.

This document is the result of extended discussions within the Semantic Web Deployment Group as a whole. Working Group members not already mentioned include (in alphabetical order): Ben Adida, Diego Berrueta, Jeremy Carroll, Michael Hausenblas, Elisa Kendall, Vit Novacek, Jon Phipps, Clay Redding, Daniel Rubin, Manu Sporny, and Ralph Swick.

Public comments (especially via the public-esw-thes@w3.org mailing list) from the following individuals provided invaluable guidance, suggestions and corrections: Mark van Assem, Stephen Bounds, Dan Brickley, Johan De Smedt, Stella Dextre-Clarke, Alasdair Gray, Andrew Houghton, Simon Jupp, Carl Mattocks, Emma McCulloch, Mikael Nilsson, Alan Ruttenberg, Aida Slavic, Simon Spero, Doug Tudhope, Bernard Vatant, Jakob Voss, Leonard Will, Sue Ellen Wright.

Appendix. Correspondences between ISO-2788/5964 and SKOS constructs

SKOS owes much to decades of efforts in the KOS community, in the form of applications, guidelines and standard formats. The compatibility between the SKOS model and two such efforts, ISO 2788 specifications for monolingual thesauri [*ISO-2788*] and ISO 5964 specifications for multilingual thesauri [*ISO-5964*] was specifically raised as a candidate requirement in the *SKOS Use Case and Requirements* [*SKOS-UCR*].

SKOS does not itself specify rules on how to create concept schemes; however, its data model reflects some KOS construction principles. The design of its vocabulary has also been especially influenced by standard thesaurus guidelines, as these are among the most mature proposals in the KOS field. In particular, there are many common points between SKOS and ISO 2788/5964. The following table summarizes the parallels and highlights ways in which the design of SKOS varies from ISO recommendations. It is hoped that this will help future efforts to port thesauri that follow the ISO guidelines into SKOS.

The reader should be aware that this comparison must not by any means be interpreted as a limitation of the scope of SKOS to standard thesauri. As already said in this document, SKOS can be used—possibly with appropriate extensions—for other types of KOS, or thesauri that do not follow the ISO guidelines.

KOS design aspect	**ISO 2788/5964**	**SKOS**
concepts vs. terms	In ISO standards, thesauri are indexing languages which consist	*Concepts* are the central modeling primitive of SKOS. Terms in ISO standards correspond to *labels* of

	of *terms.* ISO 2788 discusses extensively the crafting of terms, focusing for instance on their form. For example, explicit qualifiers are used to distinguish homographs, e.g. `Crane (bird)` vs. `Crane (lifting equipment)`.	SKOS concepts. SKOS, as a simple publishing vehicle, does not propose rules on label design. Further, since SKOS uses simple literals to represent labels, it is not possible to express term-forming mechanisms such as qualification formally and explicitly. For this, and for other cases of attaching information to labels and **not** to the concept they express, the SKOS-XL extension must be used (see Section 4.3).
intra-KOS semantic relationships — equivalence	Terms can be semantically equivalent. They are then distinguished between *preferred* and *non-preferred*, using the `USE` and `UF` (used for) relations. It is assumed that a non-preferred term can only point to one equivalent preferred term, the latter being the main entry point for the concept they both express.	Equivalent terms are represented as labels attached to a single concept. By default, there is no direct relationship between these labels. As in ISO 2788, *preferred* labels are distinct from non-preferred (*alternative*) ones. However, SKOS further allows to distinguish *hidden labels*. A concept can have only one preferred label (per language). Inside a same concept scheme, different concepts can however share a preferred label, though this is not recommended.
intra-KOS semantic relationships — other links	Beyond the equivalence relations `USE` and `UF`, three types of link are used to semantically relate terms. `BT` (broader term) and `NT` (narrower term) express that a term's meaning is more general than another's. `RT` (related term) is used when a (non-hierarchical) associative link holds between meanings, which can be useful for applications which exploit the thesaurus. ISO 2788 separates three kinds of `BT`/`NT` by	`skos:broader`, `skos:narrower` and `skos:related` mirror `BT`, `NT` and `RT` at the level of concepts. However as SKOS has a wider scope in terms of KOS types, it does not make any recommendation as precise as in ISO 2788 on what is a valid hierarchy. It is mostly up to the KOS publishers to ensure that the links in their schemes will not conflict with what is observed in general KOS practice—of which thesauri are only part. SKOS instead focuses on separating explicitly asserted "parent-child" links (`skos:broader`) from more-general "ancestor-descendant"

	means of logical tests: generic (class-species), whole-part and class-instance. If necessary, the abbreviations `BTG`, `BTP` and `BTI` can be used to represent them. The validity of logical tests in well-formed thesauri leads to transitive interpretations of the hierarchy, for which a term can reasonably admit all its ancestors as superordinates.	links which can be automatically inferred from them (`skos:broaderTransitive`) SKOS also allows for specializing semantic relationships (see Section 4.7). It does not, however, propose a standard set of such specializations. Rather, it is expected that these will come from other standards and guidelines, such as ISO 2788 itself.
syntactical composition of terms	ISO 2788 features equivalence relations that link terms to combinations of other terms (`USE +`, `UF +`), as in `coal mining USE coal + mining`.	By default, SKOS does not feature one-to-many concept-to-concept or concept-to-label links. Extensions might be however devised to address this shortcoming, e.g. by specializing `skos:Concept` or `skosxl:Label`.
node labels	Thesaurus *arrays* play an important role regarding the rendering of term hierarchy in a *systematic* display. They are for example the main vehicle for faceted organization of thesauri.	SKOS allows the representation of groupings of concepts. But it focuses on the conceptual level, and no construct is given that biases towards a specific display strategy. As a result, collections in SKOS are not explicitly related to one "parent" concept. This link must be (re-)created via a specific display algorithm, or by using an ad-hoc extension.
documentation notes	ISO 2788 proposes to attach scope notes and definitions to terms using the `SN` abbreviation.	SKOS has more types of note for concepts: scope notes, definition, history note, etc. These properties can be further extended to match specific requirements.
notations	ISO guidelines target standard thesauri. As a result, they do not address the issue of notations as used in other types of KOS.	There are two ways to attach represent notations: either via the `skos:notation` property, or by using simple labeling properties (see Section 4.6).
concept schemes	In ISO 2788, there is no explicit rendering of thesauri themselves, as	SKOS is influenced by the possibility of having several KOSs co-exist. A `ConceptScheme` class is

	terms are only considered in the context of one indexing vocabulary.	proposed to represent them explicitly and to attach descriptive metadata to them, even though SKOS itself does not feature specific constructs for this. The link between a KOS and its concepts is explicit, and a same concept can belong to several KOSs.
top concepts	In a thesaurus display, the `TT` abbreviation can be used to refer to the topmost term of the hierarchy to which displayed terms belong.	`skos:hasTopConcept` is used to relate a concept scheme to the concepts that constitute entry points in its hierarchy.
language management	In ISO 2788 terms should come from a same language. ISO 5964 proposes to have several languages co-exist in a same thesaurus. The terms from each language form however quite independent parts of the thesaurus, only related to each other by *translation* links.	From a model perspective, concepts are language-independent : a concept can have labels in different languages. Labels can indeed be declared as language-specific, using RDF literal language tags. Several languages may therefore be tightly integrated in a same concept scheme.
inter-KOS mapping relationships	Semantic mapping relations are only considered by ISO 5964 in the context of multilingual thesauri, as a further characterization for the translation. The types discussed are: • exact equivalence, • inexact equivalence—terms express a same general idea but their meaning is not fully identical, • partial equivalence—the meaning of one term is broader than another's,	SKOS mapping relations mirror relatively well ISO 5964 types. For example, `skos:exactMatch` and `skos:closeMatch` separate cases where equivalence is perfectly valid from a semantic perspective from other cases where semantic equivalence is not exact but can be accepted for a given application. For an individual multilingual KOS, however, equivalence links in ISO 5964 may be represented in SKOS by attaching equivalent terms as labels of a same concept. This fits the approach of ISO 5964, which only makes it necessary to link preferred terms: such links can be transferred at the level of the concepts these terms express. Yet

	• single-to-multiple—a concept expressed by one term in the source language is expressed by a combination of terms in the target language. Note that ISO 5964 addresses many issues that are outside the scope of SKOS, such as transferring hierarchical and associative relations from one language to the other, or coining new terms in a language when a semantic equivalent cannot be found for terms in other languages.	ISO 5964 also allows to relate non-preferred terms (e.g., `"DNA"@en` and `"ADN"@fr`). In SKOS, such links can be represented only using the SKOS-XL extension. Single-to-multiple translations cannot be represented in SKOS. As for syntactic combination of terms within one thesaurus, extensions to the standard model are required. Note finally that ISO 5964 discusses extensively the display of multilingual thesauri. SKOS does not address this. But as for simple thesauri, ISO 5964 displays can be implemented on top of SKOS data—except in the case of the single-to-multiple mappings mentioned above.