DRAFT DISCUSSION PAPER
NSF/NSDL Metadata Registry Project
September 23, 2006

For DC-Registry Discussions at DC-2006 in Manzanillo, Mexico

The Notion of the "Concept Instance": Problems in Modeling Concept Change in SKOS

Joseph Tennis <jtennis@interchange.ucb.ca>
Stuart Sutton <sasutton@u.washington.edu>
Diane Hillmann <dih1@cornell.edu>

The U.S. National Science Foundation metadata registry under development for the National Science Digital Library (NSDL) is a repertory intended to manage both metadata schemes and schemas. The focus of this draft discussion paper is on the scheme side of the development work. In particular, the concern of the discussion paper is with issues around the creation of historical snapshots of concept changes and their encoding in SKOS. Through framing the problem as we see it, we hope to find an optimal solution to our need for a SKOS encoding of these snapshots. Since what we are seeking to model is concept change, it is necessary at the outset to make it clear that we are ***not*** talking about changes to a concept of such a nature that would require the declaration a new concept with its own URI.

In the project, we avoid the use of the terms "version" and "versioning" with regard to changes in concepts and reserve their use to the *significant* changes of schemes as a whole. Significant changes triggering a new scheme version might include changes in scheme documentation that express a significant shift in the purpose, use or architecture of the scheme. We use the term "snapshot" to denote the state of a scheme at identifiable points in time. Thus, snapshots are identifiable views of a scheme that record the incremental changes that have occurred to concepts, relationships among concepts, and scheme documentation since the last snapshot. Aspects of concept change occur that we need to capture and make available both through the registry and through potentially in transmission of a scheme to other registries. We call these capturings "concept instances."

The problem space we need to address is not unlike the problem faced by DCMI in the documentation of schema terms in its various namespaces. Terms in the DCMI namespaces have changed over time as they have been refined for clarity and other purposes. Those changes have been captured in DCMI documentation. For example, see *DCMI Terms: A Complete Historical Record* (http://www.dublincore.org/usage/terms/history/) where terms have been provided "version" numbers (e.g., http://dublincore.org/usage/terms/history/#contributor-003 for the third version the DCMI "contributor" property). What this implies is that our understanding of the current state of a DCMI property is operationally a function of the URI for the *abstract* entity identifying the

concept (e.g., http://purl.org/dc/elements/1.1/contributor) and the values used to describe the concept expressed in the most recent version of the property (http://dublincore.org/usage/terms/history/#contributor-004).

Therefore, the DCMI property "contributor" conceptually consists of an abstract entity (identified by URI) that came into being 6 August 1998 that was *made manifest* in version one and revised through subsequent revisions in 1999 and twice in 2002. This makes up one abstract notion (contributor) and four physical manifestations of that notion over a period of five years. While the structure of the DCMI documentation obscures this conceptual framework, we see no other way of explaining the phenomenon.

We posit that the notion of a DCMI property is somewhat akin to the notion of a "work" in the IFLA *Functional Requirements for Bibliographic Records* (FRBR). (http://www.ifla.org/VII/s13/frbr/frbr.pdf)

> "A *work* is an abstract entity; there is no single material object one can point to as the *work*. We recognize the *work* through individual realizations or *expressions* of the *work*, but the *work* itself exists only in the commonality of content between and among the various *expressions* of the *work."* (FRBR, page 16)

So, we would assert that we recognize the DCMI term by means of its individual expressions/versions and that it actually exists in what is common among all of the expressions/versions. (An exception to this statement might be a versioning of a term to correct errors. In many contexts, such non-substantive corrections would not be considered versioning.)

In like fashion, the SKOS notion of a "concept" is defined as an abstract entity—"An abstract idea or notion; a unit of thought." (http://www.w3.org/TR/2005/WD-swbp-skos-core-spec-20051102/). So defined, the SKOS abstract concept can exist in many different schemes. However, unlike FRBR, SKOS provides no separate entity to represent the physical manifestation of the abstract concept. So instead of some *particular* manifestation of the concept existing in a *particular* scheme, the abstract concept is treated as the only physical manifestation. We think this is the crux of our problem in trying to model concept change in SKOS. Since all changes over time to property values for a concept manifestation exist alongside the properties defining the relationship between the concept and its potentially many versions through "inScheme," there is no intrinsic means to associate a particular expression of the concept with a particular scheme. We provide a simple illustration of our problem in Figure 1 where we see a single SKOS concept existing in many schemes and also reflecting various changes (simply identified here with multiple change notes).

In our exploration of the problem, our proposed notion of a "concept instance" is somewhat analogous to the IFLA "manifestation" by which a specific expression of a work in the IFLA model is given tangible form. Thus, in our exploration, an initial recognition of an abstract SKOS concept is by means of its initial manifestation—in other words, through the first "concept instance." As in the FRBR model, the concept actually "exists" only in the

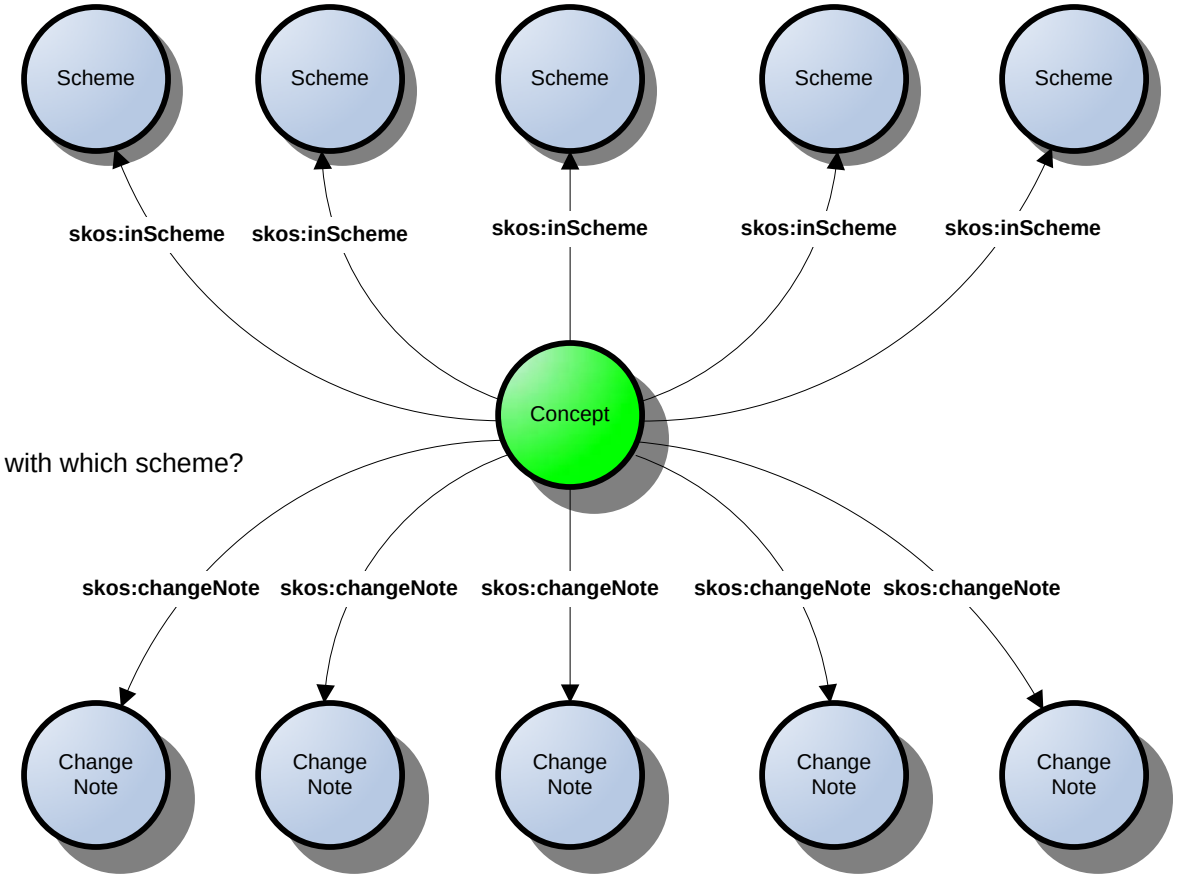
commonalities among its various instances. We provide a simple illustration of this new entity representing the particular manifestation of the abstract concept in Figure 2.

Figure 3 illustrates our first statement of the fuller result of providing an intermediate entity representing the physical instantiation of a particular abstract SKOS concept between that concept and the various schemes in which it exists. Each concept instance in Figure 3 is a full expression of the concept semantics just as each of the DCMI term versions expresses the full semantics of the term (see http://www.dublincore.org/usage/terms/history/). While DCMI does not formally assert that its version identifiers are URIs, we do so assert that each of the concept instances is identified by its individual URI. As a result, we are able to associate each identifiable manifestation of the abstract concept with an associated scheme. Figures 4 through 6 provide various visual explanations of our thinking.

In figures 7 and 8, we provide flow charts describing scheme snapshots and versions and, by inference, the concept change processes we have described here.

FIGURE 1:

- SKOS postulates that a concept exists independent of a scheme
- Therefore, a concept can exist in many schemes (inScheme)
- However, each scheme delimits the meaning of a concept by its relationships with other concepts in the scheme
- Change notes, as properties of a concept, are not linked to the scheme in which the change applies



- Which change note goes with which scheme?

FIGURE 2:

General Solution:
Concept “Instances”

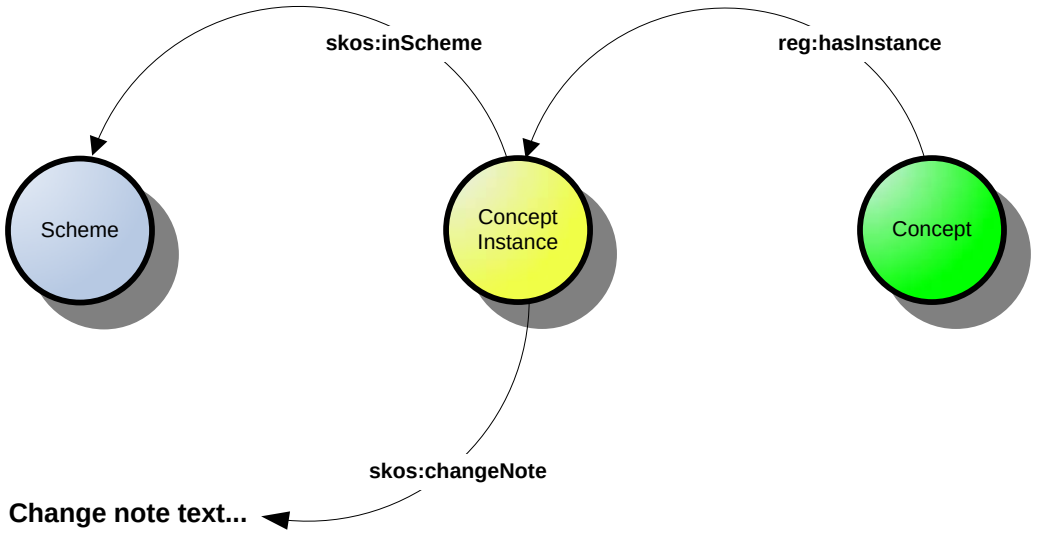


FIGURE 3:

Solution Example

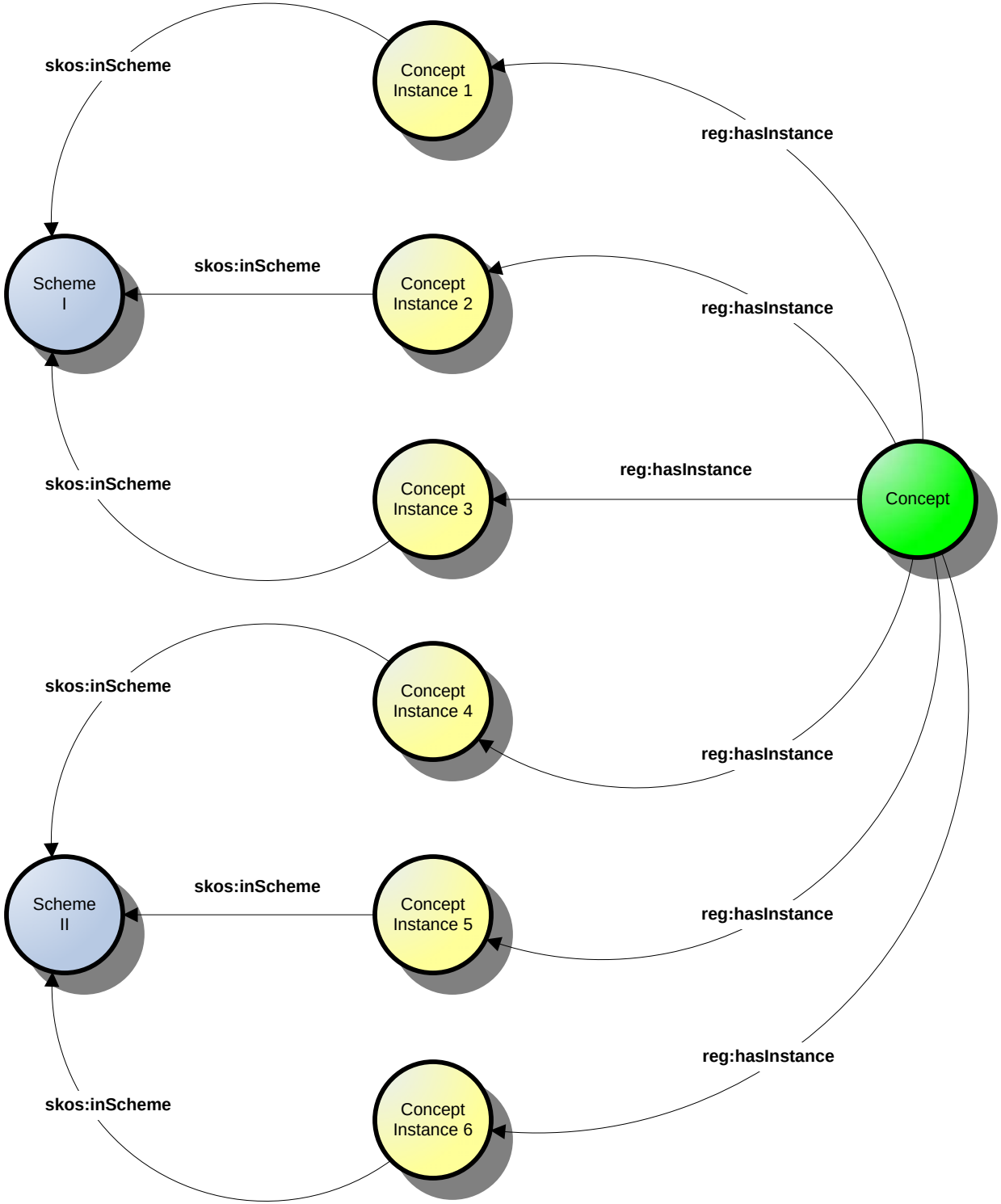


FIGURE 4:

Example Concept Change--Lumping

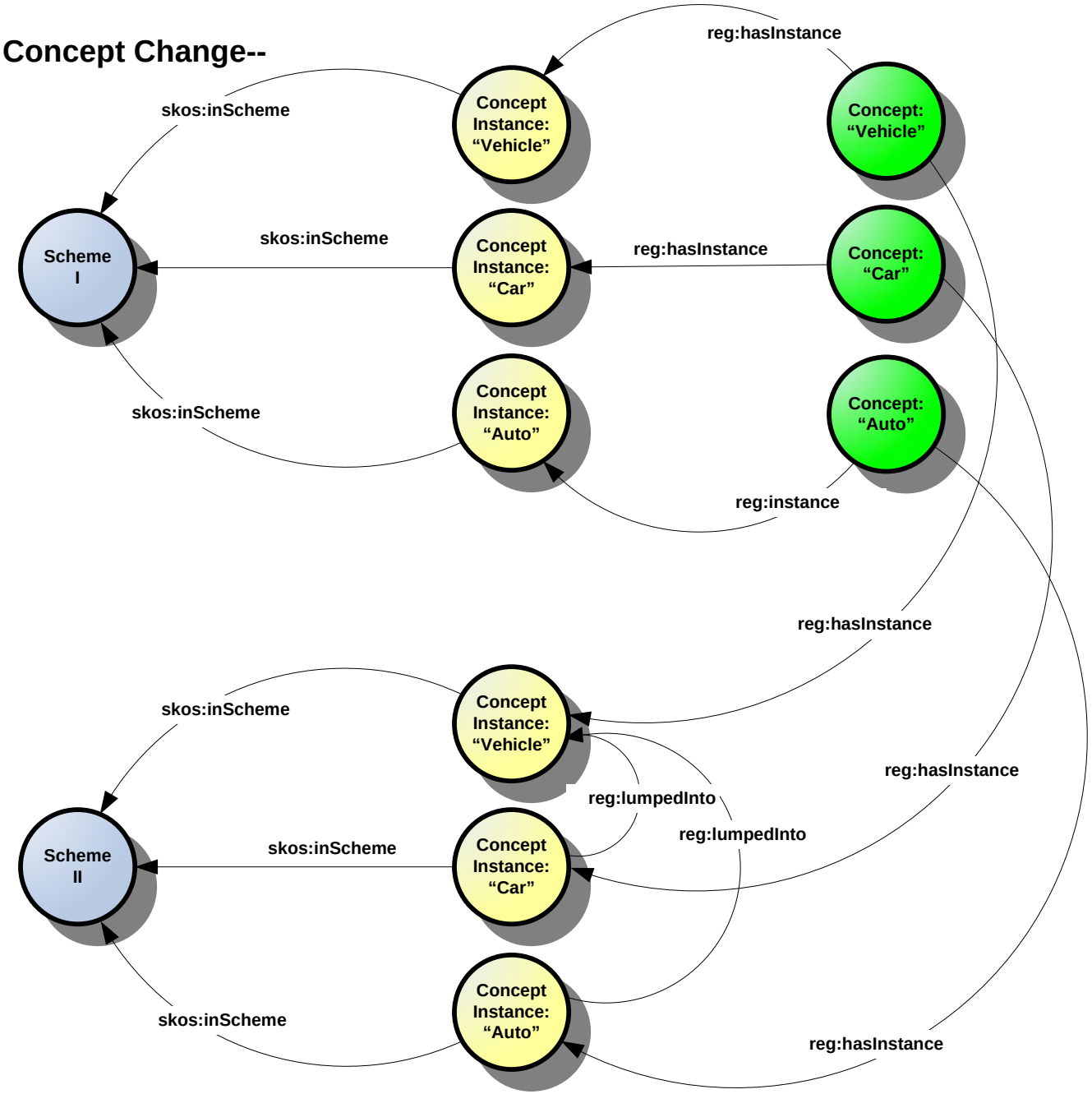


FIGURE 5:

Instance Snapshots

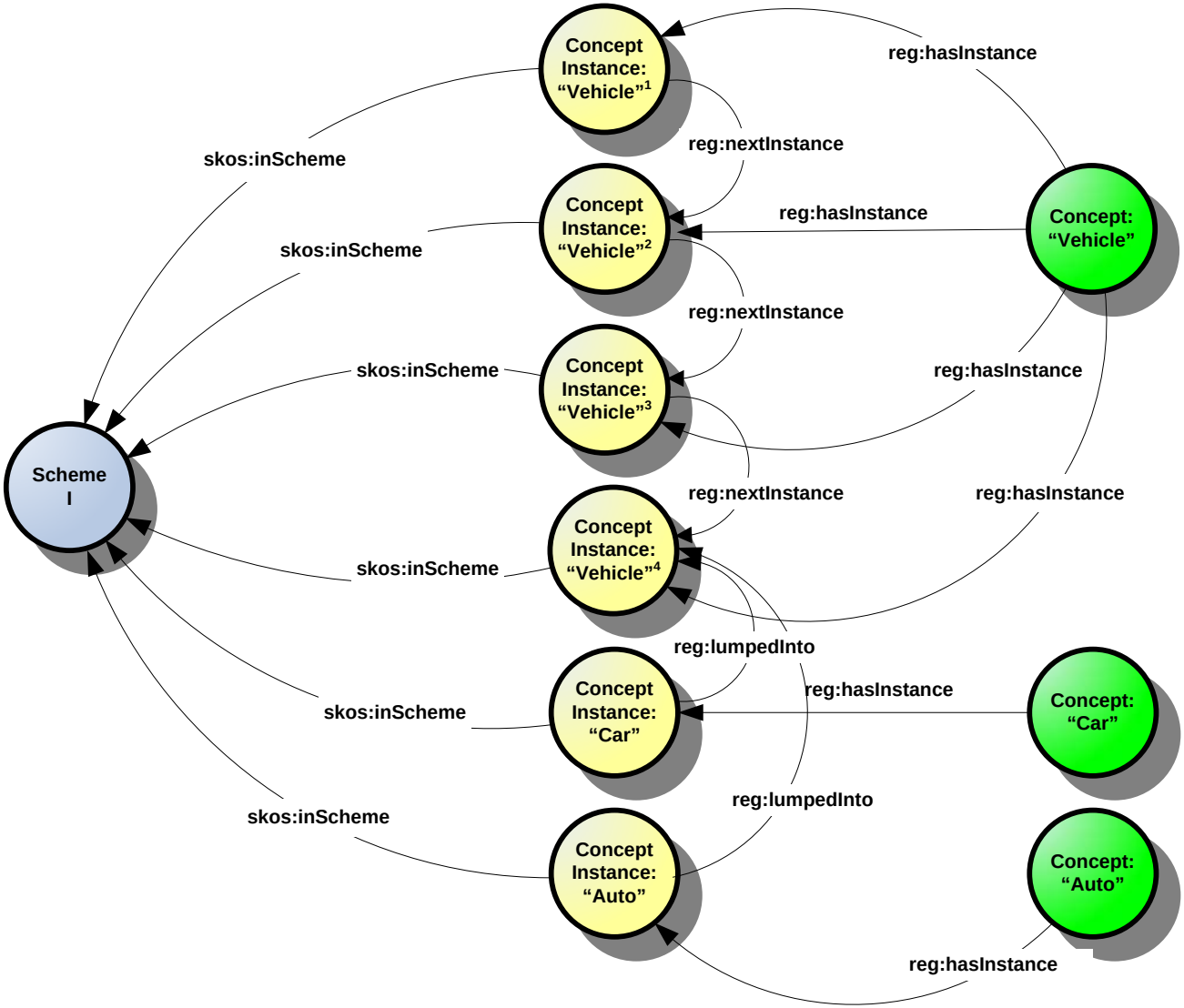


FIGURE 6:

Major Relationships Among Concepts, Concept Instances, Schemes and Select Associated Properties

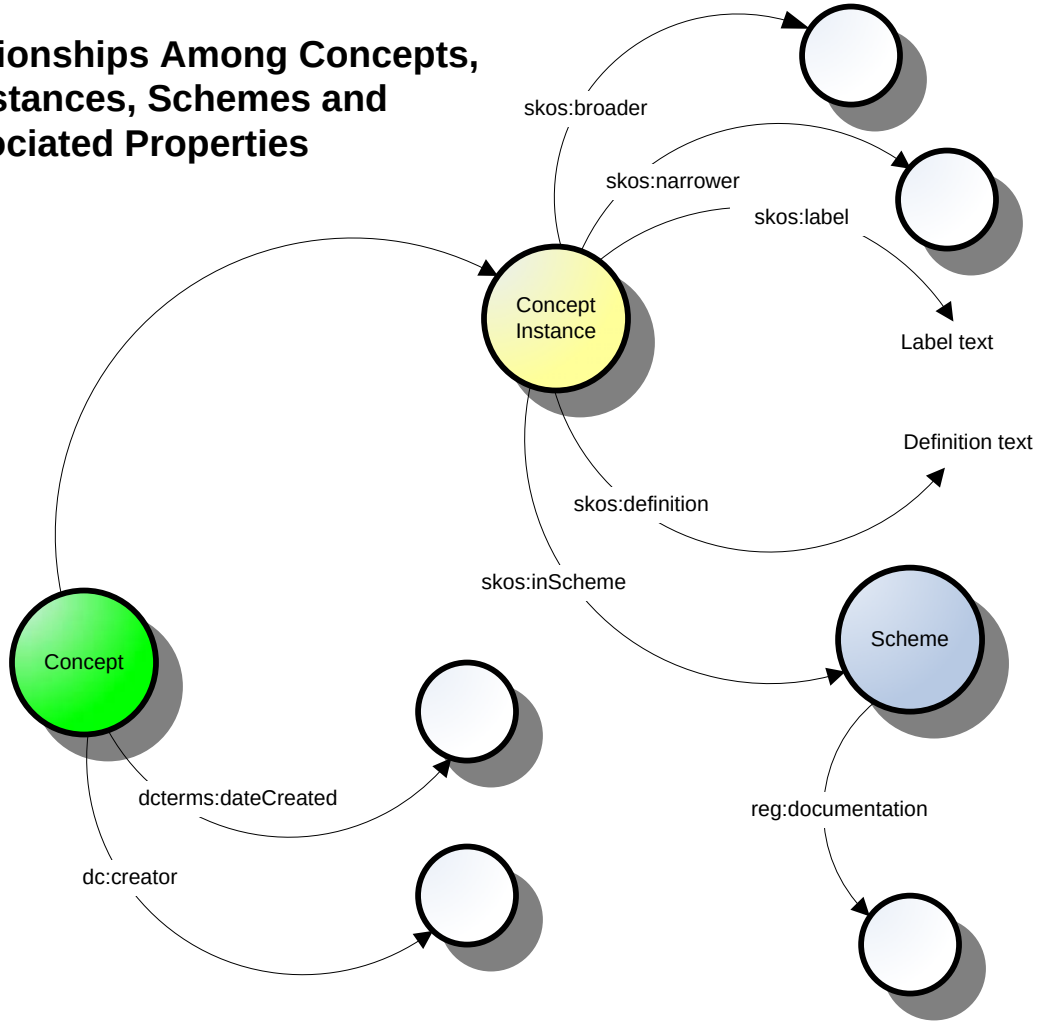
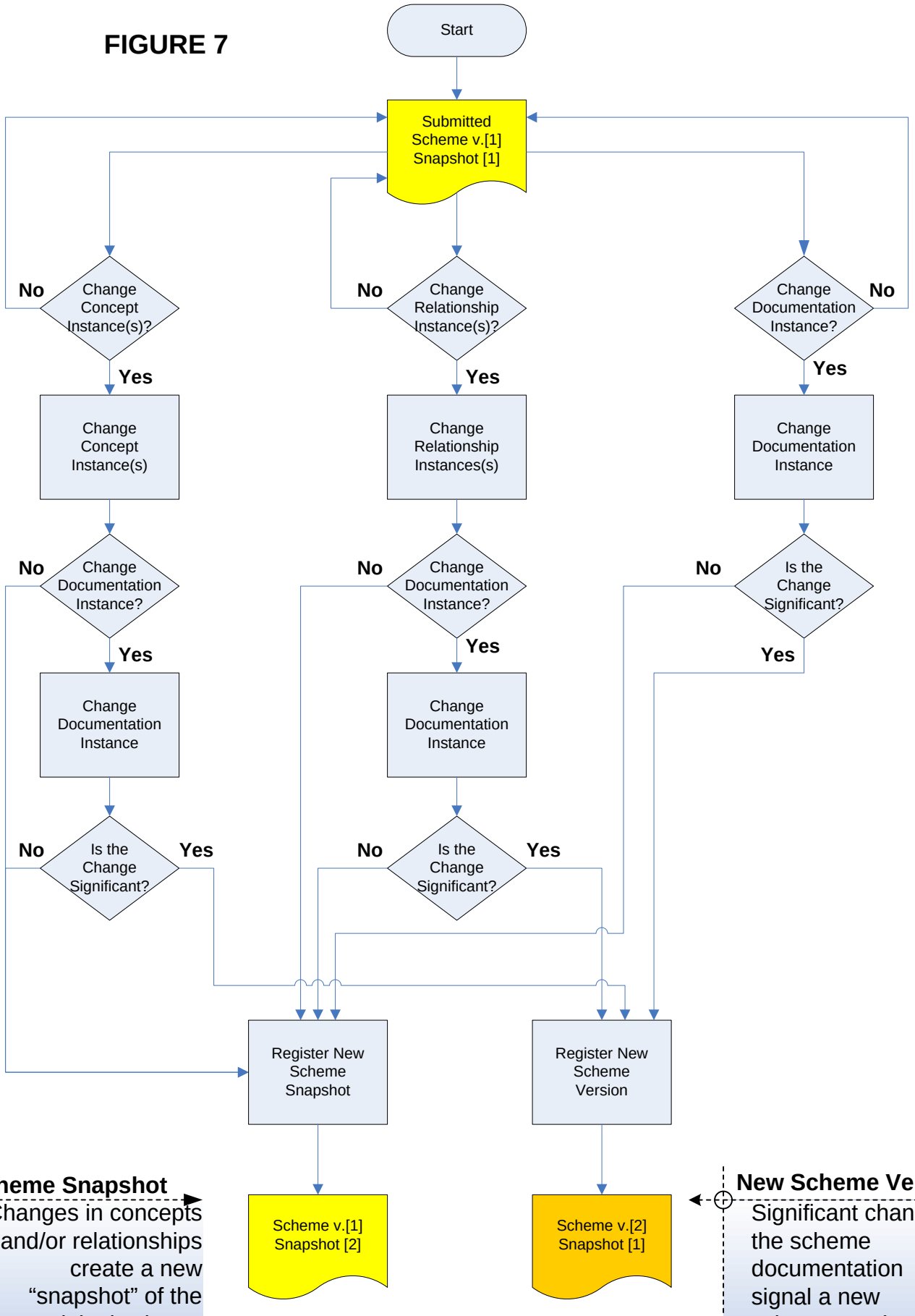


FIGURE 7



Scheme Snapshot

Changes in concepts and/or relationships create a new "snapshot" of the original scheme

New Scheme Version

Significant changes in the scheme documentation signal a new scheme version

Registering Scheme Snapshots and Scheme Versions

FIGURE 8:

Registering Concepts, Relationships and Scheme Documentation