Ontology Repositories: A Treasure Trove for Content Ontology Design Patterns

Torsten HAHMANN
National Center for Geographic Information and Analysis, School of Computing and Information Science, University of Maine, Orono, ME, USA

Abstract. Ontology design patterns (ODPs) are widely accepted as important tools for accelerated design of ontologies. We revisit content patterns (CP), an important class of ODPs, and distinguish two kinds based on their degree of formalization and maturity: conceptual CPs and formalized CPs. We show how formalized CPs and the closely related knowledge patterns have natural equivalents in modular ontology repositories. Common notions of pattern reuse (including specialization, instantiation, and composition) are also expressible as logical relationships in the repository. Thereby, ontology repositories support identifying mature formalized CPs and knowledge patterns and support documenting the patterns' reuse.

Keywords. ontology engineering, ontology design patterns, content pattern, ontology repository, modularity, hierarchy

1. Introduction

Design patterns can be thought of as general guidelines—"best practices"—that are modular, widely reusable, and that address frequently encountered design problems in a particular domain, such as architecture, software engineering, or ontology engineering. Design patterns often take the form of templates, which capture key aspects of solving a common problem while leaving room for customization to the particular problem context. Such modular components of larger systems are commonly used throughout all engineering disciplines to reduce the complexity of large systems by providing standardized solutions to commonly occurring problems.

Ontology design patterns (ODPs) are modular, reusable pieces for ontology engineering. Most closely related to the notion of ODPs are software design patterns [3], which provide solutions to standard problems in software design. Software design patterns typically address structural problems in object-oriented design: how to solve a reappearing problem through the use of a certain program structure, such as the introduction of a special-purpose class, e.g. a singleton, a factory, or a mediating class, or the introduction of a set of classes coupled in a certain way, e.g., through inheritance.

Since ontology engineering has strong parallels with object-oriented software design in its focus on concepts (classes) and relations (methods, which are functions and thereby relations as well), one can expect ontology design patterns to serve in a similar role in ontology design as software design patterns in software design. However, a peculiarity in ontology engineering is that the term ontology design patterns as coined by [4] not only encompasses *structural solutions* to common ontology engineering challenges, but also much more concrete solutions in the form of small, reusable pieces of ontologies referred to as *content ontology design patterns* (abbreviated in the following as CPs). CPs are quite distinct from other kinds of ODPs used in ontology engineering, such as *structural design patterns* (which are either logical and architectural ODPs [2]). While structural ODPs offer template solutions much like software design patterns, CPs are chiefly concerned with formalizing a particular piece of knowledge. Over time, CPs have become the predominant kind of ODPs as demonstrated by the number of submissions in the different categories of ODPs on the website `ontologydesignpatterns.org`, the most widely used repository for ontology repositories. The category of CPs received by far the most submissions, significantly more than *structural ODPs*, the second-most popular category[1].

In this paper, we reexamine three different kinds of CPs discussed in the literature and study to what extent they satisfy the general requirements of ODPs from [4]. Furthermore, we show how CPs are related to particular kinds of modules in modular ontology repositories as introduced by [8] and [6], thereby being more explicit about the nature of the close relationship between certain kinds of ODPs and ontology repositories, a line of inquiry initiated in [12]. In particular, we identify what modules in an ontology repository qualify as specific kinds of CPs and outline how an ontology repository can help extract mature CPs and document their use—for example through extension or instantiation—through formal logical relationships within the repository. This gives a more general view of how modular ontology repositories can play a pivotal role in the identification, formalization, documentation, and reuse of CPs in the future, overcoming the current bottleneck of identifying and formally documenting mature and reusable CPs. More specifically, our contributions are the following:

1. We distinguish three kinds of CPs: *conceptual CPs*, *formalized CPs*, and *knowledge patterns* [1] and explain how they differ.
2. We show that the inherent nature of conceptual CPs is incompatible with the requirements of what constitutes an ontology design pattern proposed by [4].
3. We draw a strong parallel between formalized CPs and certain modules from ontology hierarchies. Particularly, we show that all root modules (or root theories) of *generic ontology hierarchies* in the sense of [6] qualify as formalized CPs. Similarly, arbitrary modules from *mathematical ontology hierarchies* in the sense of [6] qualify as knowledge patterns. This reconciles two competing views of ontology engineering: (1) ontology engineering through the reuse of modules from an ontology hierarchy and (2) pattern-based ontology engineering as recombination, extension, and instantiation of CPs.

The immediate consequences of these contributions include the following three:

(a) As an immediate consequence of 3., we suggest CPs to be harvested effortlessly from existing modular ontology repositories. The harvested CPs satisfy all criteria of ODPs and are more mature than many currently available CPs.

(b) Moreover, ontology repositories maintain a formal documentation of uses of these patterns through logical intertheory relationships, filling the need for documentation and exemplification demanded by [17].

(c) As an ancillary consequence, we face an important choice in future work: either accept a surge in CPs, or identify additional conditions that a root module of a generic ontology hierarchy must satisfy in order to qualify as a formalized CP.

[1] `ontologydesignpatterns.org` lists 101 CPs and 14 structural ODPs as of April 3, 2014.

The paper is structured as follows. Section 2 reviews the notion of content ontology design patterns (CPs) from the literature and distinguishes two kinds of CPs: conceptual CPs and formalized CPs. Section 3 studies conceptual CPs in more detail, identifying three key properties usually associated with ontology design patterns that conceptual CPs lack. Section 4 studies formalized CPs in-depth, relates them to generic hierarchies as defined for ontology repositories, and shows why knowledge patterns [1] are a mathematical variant of formalized CPs closely related to the mathematical hierarchies from ontology repositories. Background and terminology concerning ontology repositories are introduced in Section 4 as needed. Section 5 discusses how this close relationship between ontology repositories and ontology patterns can be exploited for extracting and documenting patterns and for pattern-based ontology design.

2. Content ODPs

Early on, the special role of CPs as solutions to content problems instead of structural problems has been acknowledged in the literature [4, 14]. One key difference between CPs and other kinds of ODPs is how they are used in the ontology design process. Structural ODPs, for example, offer a skeleton of a solution that are populated with formalized content (usually as a set of axioms in an ontology language) in order to model a specific aspect of some domain. In contrast, CPs offer a specific conceptual piece of knowledge that is encountered across different specialized domains. This may include generic spatial and temporal knowledge such as generic facts about events, processes and changes. CPs come equipped with specific axioms (sometimes rather informally specified in a conceptualization) capturing such generic, though those axioms may, on a case-by-case basis, be supplemented by domain- or application-specific axioms. Often, such CPs can be simply glued together in certain ways to design an ontology about a larger domain without having to design novel modules from scratch. The remainder of this paper focuses exclusively on CPs and disregards other types of ODPs.

CPs were originally conceived as formalizing a *generic use case* (GUC) that capture a recurrent conceptual piece about a certain reusable domain and that address a limited set of competency questions [7] about that particular domain. We call such conceptual pieces that lack an axiomatization in a formal ontology language *conceptual CPs* and distinguish them from *formalized CPs*, which provide specific axiomatizations of conceptual CPs. The original idea of CPs seems to encompass both formalized CPs and conceptual CPs [4, 14], but we believe that it is beneficial to explicitly distinguish them because of their different properties and their roles at different stages within the ontology design process. Of course, the two kinds of CPs are closely related: each formalized CP is grounded in a conceptual CP and, reversely, each conceptual CP gives rise to one or multiple formalized CPs, which should only differ in the extent of the formalization determined by the expressiveness of the chosen formal language. Designing a conceptual CP is a first step towards defining a formalized CP, which can be thought of as a specific piece of ontology, that is, a "mini-ontology" or an ontology module, that extracts a certain piece from a foundational or core ontology [4].

Next, we study both conceptual and formalized CPs in more detail. We will argue that the conceptual variant is insufficiently formal to satisfy the requirements imposed on ODPs by [4, 14]. Subsequently, we show how formalized CPs are related to modules defined by the hierarchical structure of an ontology repository and how they can be extracted from such an ontology repository with ease.

3. Conceptual CPs

Conceptual CPs are essentially conceptual models that capture a specific set of closely linked concepts, its interrelationships, as well as its relationships to other concepts outside the restricted realm of the conceptual model of interest. Multiple conceptual patterns can be composed to model a more complex domain or system [14].

As already pointed out by [2], conceptual CPs are used at the early stage of the ontology development cycle, namely when analysing the—generic or specific—domain and creating a conceptual model thereof. Due to their role in the early stages of ontology development they lack a full formal axiomatization and thus cannot be readily reused as a module (reuse by extension/specialization) or as a template (reuse by instantiation/cloning). Instead, developers that encounter a similar content problem can only rely on the conceptual analysis and develop their own formalization thereof. Because such conceptual CPs are the product of early stage development, they lack key properties typically associated with ontology design patterns (compare [2,4]):

- they lack sufficient formalization for easy reuse as an extensible module or an instantiable template,
- they lack documented instantiation in fully formalized and practically used ontologies that prove reusability,
- they lack sufficient maturity to be called “best practices”.

Instead, a conceptual pattern is an initial attempt at formalization that has not yet matured enough to be fully formalized. For this reason, calling such a conceptual model a pattern is premature, it only becomes a pattern once it is formalized, used, and matured[2]. Before this happens, they are more appropriately called *conceptual modules* or, as originally in [4], *generic use cases* (GUCs). Many recently proposed conceptual CPs, including those of *Semantic Trajectory* [10] or *Surface Water* [16], are initial attempts of creating a conceptual model. Thereby, they are significantly different from how CPs were envisioned to be created [14]: all methods discussed in [14] presuppose the existence of a conceptual model—ideally already formalized in an ontology language—for extracting a CP.

Note that not all conceptual CPs lack in all three criteria to the same degree. For example, the conceptual CP *Semantic Trajectory* [10] is at least formalized in OWL-DL (though the language’s expressivity is rather restricted compared to full first-order logic), but has not been around long enough for it to be tested, matured, and documented through actual use “in the wild” [10]. The large number of concepts and relations of both the conceptual CP *Semantic Trajectory* and the conceptual CP *Surface Water* [16]—each consisting of at least 10 classes and 14 relations depending on how one counts—may seriously inhibit reusability. Highly reusable patterns are typically much smaller.

4. Formalized CPs

Our second perspective treats CPs as theories formalized in a specific formal language. Such formalized CPs can be linked via the concepts and relations in their signatures that cross boundaries between two related CPs. Larger patterns can be built by taking the union of the specific formalizations and adding formal statements (in the same

[2]While this notion of a conceptual pattern does not match the common definition of an “ontology design pattern”, it is still a pattern in the broader, more general sense of the word as used, e.g., in “pattern matching” because such a conceptual pattern appears repeatedly across different applications and/or domains.

formal language) about how the individual CPs are glued together (compare the composition of CPs as discussed in [14]).

Formalized CPs are thus restricted to a specific ontology language; they are no longer language-independent. However, it is still possible to have multiple formalized CPs that encode a particular conceptual piece of knowledge. In this case multiple formalized CPs grounded in a shared conceptual CP are feasible. Alternatively, one can formalize the conceptual CP in the most expressive formal language of interest, such as first-order logic, and extract lightweight versions in less expressive formal ontology languages as needed. In either case, a formalized CP is a module of a larger reference ontology. Since modules are also thought of as reusable pieces of a larger ontology, the question of adequate modularization arises in the context of formalized CPs: how can we identify candidates for formalized CPs that satisfy the desired properties of encapsulation and reusability from an ontology repository? Simply treating all modules as CPs results in a too large set of CPs, for which reusability is not ensured. Instead, we want to extract special modules that are guaranteed to be maximally reusable across multiple domains and applications.

In this section, we show how modular ontology repositories provide intrinsic mechanisms that support the extraction of maximally reusable formalized CPs. To make this point, we use a view of ontology repositories from [6, 8, 9] as a set of ontologies that are grouped into hierarchies based on their primitive signature (the set of undefined concept and relation symbols). Ontologies within a hierarchy use the same primitive language and are organized by nonconservative extensions. The hierarchies themselves are partially ordered by reducibility relationships among them. First, we will review the core ideas necessary to follow this view and then show how this structure is a basis for a logically-founded distinction between mathematical and generic ontologies [6]. We then show how formalized CPs naturally fit into this view by playing the role of root modules of generic hierarchies. So-called knowledge patterns [1] also fit into this view by playing the role of modules from mathematical hierarchies.

4.1. Ontology Repository: A Partially Ordered Hierarchy of Ontology Hierarchies

We follow the approach to modular ontology repositories taken in [8], with some more recent adaptation in [9][3]. The most fundamental logical intertheory (i.e. between ontologies) relationships of interest are conservative and nonconservative extensions and faithful and relative interpretations. One theory *extends* another one if it preserves all theorems. Such an extension is *conservative* if, and only if, all new theorems involve new primitive symbols (i.e. new concepts or relations), otherwise it is *nonconservative*. *Relative interpretations* generalize extensions in that the involved theories have disjoint signatures, but a translation from the interpreting theory's signature into the interpreted theory's signature exists such that the translated theorems are preserved. Such a relative interpretation is *faithful* if, and only if, the interpreting theory does not entail new theorems whose translation is not entailed by the interpreted theory[4].

[3]We follow [8] in terminology and notation, treating ontologies and their modules as logical theories. We do not distinguish between logically equivalent theories. For every theory T, $\Sigma(T)$ denotes its signature, which includes all the constant, function, and relation symbols used in T, and $\mathcal{L}(T)$ denotes the language of T, which is the set of first-order formulae that only use the symbols in $\Sigma(T)$. We assume that for every theory T a special primitive signature $\Lambda(T)$ exists. Such a primitive signature is a distinct minimal subset of the signature in which all other nonlogical symbols in $\Sigma(T) \setminus \Lambda(T)$ are definable.

[4]See [9] for the full definitions and more detailed explanations.

Two ontologies that have equivalent primitive signatures, that is, that each have a set of primitives that are equivalent (up to symbol renaming), are said to be in the same ontology hierarchy. This considerably strengthens the definition of a hierarchy as containing only ontologies with identical signatures from [8]. We can compare two ontologies T_1 and T_2 in the same hierarchy, by checking whether one of them is *more restricted* than the other, meaning that the latter one is a nonconservative extension of the former one, written as $T_1 < T_2$[5].

Definition 1. *[9] An ontology hierarchy* $\mathbb{H} = \langle \mathbf{H}, \leq \rangle$ *is a partially ordered, finite set of theories* $\mathbf{H} = T_1, ..., T_n$ *such that*

1. *for all* i, j *with* $1 \leq i, j \leq n$ *there exist some sets of primitives* $\Lambda(T_i)$ *and* $\Lambda(T_j)$ *such that* $\Lambda(T_i) = \Lambda(T_j)$*;*
2. $T_1 \leq T_2$ *iff* T_2 *is an extension of* T_1*;*
3. $T_1 < T_2$ *iff* T_2 *is a nonconservative extension of* T_1.

Hierarchies have special modules, called *root theories* [6,8], which are theories that capture a set of most general assumptions in the hierarchy, that is, mo root theory nonconservatively extends other theories within its hierarchy. A theory T that extends a hierarchy's root theory and that has an equivalent primitive signature is said to be *compatible with the hierarchy*[6].

Definition 2. *[8] A theory* T *in a hierarchy is a root theory iff it does not nonconservatively extend any other theory in the same hierarchy.*

In [8], we introduced the concept of a *closed hierarchy*, which is a hierarchy that has a unique root theory extended by all other theories within the hierarchy. Formally, a closed hierarchy is defined as a hierarchy closed under similarities. The similarity of two theories T_1 and T_2 within a hierarchy is their shared set of theorems that do not arise only as disjunctions of theorems of the individual theories T_1 and T_2.

Definition 3. *(Adapted from [8]) Let* T_1 *and* T_2 *be theories in the same hierarchy with the primitive signature* Λ.

The similarity between T_1 *and* T_2 *is the strongest theory (up to logical equivalence)* $S \subseteq T_1 \cap T_2$ *with* $\Lambda(S) = \Lambda(T_1)$ *so that for all* $\sigma, \omega \in \mathcal{L}_\Lambda(T_1)$ *if*

$$T_1 \models \sigma \quad and \quad T_2 \models \omega \quad and \quad S \not\models \sigma \quad and \quad S \not\models \omega$$

then either $\sigma \vee \omega$ *is independent of* S *or* $\sigma \vee \omega$ *is a tautology.*

A *repository* is then a set of ontology hierarchies, related by reducibility.

Definition 4. *[8] A theory* T *is reducible to a set of theories* $T_1, ..., T_n$ *iff*

1. T *faithfully interprets each theory* T_i*, and*
2. $T_1 \cup ... \cup T_n$ *faithfully interprets* T.

[5]Notice that the primitive signature within a hierarchy is fixed, thus the only way to properly extend a theory in the hierarchy is by further restricting the interpretation, leading to a nonconservative extension.

[6]We take a "lax" approach to hierarchies and explicitly store only theories that either have a practical use, are modules of a reduction for another theories, or arise through the explicit closure of hierarchies under similarities and differences. Compatibility with a hierarchy means that a theory *could* be stored in the hierarchy.

The reducibility relation is used to partially order hierarchies within a repository by their primitive signature (in addition to nonconservative extensions), which limits repositories to closed hierarchies with a single root theory.

Definition 5. *[8] Let $\mathbb{H}_1, ..., \mathbb{H}_n$ be a finite set of closed hierarchies.*

A repository $\mathbb{R} = \langle \mathcal{R}, \sqsubseteq \rangle$ is a partially ordered set $\mathcal{R} = \{\mathbb{H}_1, ..., \mathbb{H}_n\}$ of closed hierarchies such that $\mathbb{H}_i \sqsubseteq \mathbb{H}_j$ iff the root theory of $\mathbb{H}_j$ is reducible to a set of theories $T_1, ..., T_n$ such that at least one T_i is compatible with $\mathbb{H}_i$.

In other words, $\mathbb{H}_1 \sqsubseteq \mathbb{H}_2$ means that the root theory of $\mathbb{H}_2$ extends the root theory of $\mathbb{H}_1$. The theories in $\mathbb{H}_2$ either reuse the theories from $\mathbb{H}_1$ while imposing additional minimal conditions, or extend the primitive language of $\mathbb{H}_1$ (compare [9]). Each repository has a set of irreducible hierarchies, namely the hierarchies $\mathbb{H}_i$ for which no $\mathbb{H}_j$ with $\mathbb{H}_i \sqsubseteq \mathbb{H}_j$ exists. The root theories of irreducible hierarchies are modules closely related to formalized CPs. Fig. 1 gives an example of interrelated hierarchies within an ontology repository. Their division into generic and mathematical hierarchies is explained next.

4.2. Generic and Mathematical Hierarchies

The ontology hierarchies within the repository are partially ordered by irreducibility, but in practise we often encounter multiple hierarchies that are either logically equivalent or that share theories but differ in the axioms of their root theories. This often happens when standard mathematical structures, such as partial orders, graphs, or incidence structures, are reused in the axiomatization of several realms of generic knowledge, such as theories of space, time, events, processes, and of domain-specific knowledge.

The key difference between a mathematical and a generic theory is the absence (or presence) of an intended semantics. Mathematical structures are abstract per se—vocabulary terms (such as vertices, edges, and vertex adjacency in graphs, or the partial order relation $\leq$ in posets) have no particular meaning, i.e. grounding in the real world, they are purely abstract symbols. Non-mathematical theories, including generic[7] and domain-specific ontologies, always come with some—albeit often vague—intended meaning that corresponds to a concept or relationship in the represented (real) world. For example, no matter what generic ontology of time one uses, the terms 'time point', 'time interval', 'before', 'after', 'during', etc. have intended meanings: 'time points' and 'time intervals' are things that can be temporally ordered, while 'before' and 'after' are temporal ordering relations, not arbitrary (partial or linear) orders (compare [5]). Their meanings vary slightly depending on the specific conceptualization and ontology one uses, but they share certain semantics that make them ontologies of time. This shared semantics is inherent in the axiomatization of a generic hierarchy's root theory.

A closer investigation into differentiating criteria of generic theories (and hierarchies) has been initiated in [6]. The presence of *ontological commitments* and *ontological choices* have been identified as distinguishing generic ontologies from other ontologies. Crudely speaking, ontological commitments capture the essence of a generic concept or relation, while ontological choices capture the optional semantics that ontologies of the same generic domain may differ in. For example, stating that an ontology of time always

[7]The use of the term *generic ontology* here and in [6] encompasses what [15] call foundational ontologies and core ontologies. Loosely speaking, foundational ontologies can be thought of as irreducible generic ontologies, while core ontologies correspond to reducible generic ontologies. However, there is no clear logical distinction between core ontologies and domain ontologies; both correspond to reducible generic ontologies.

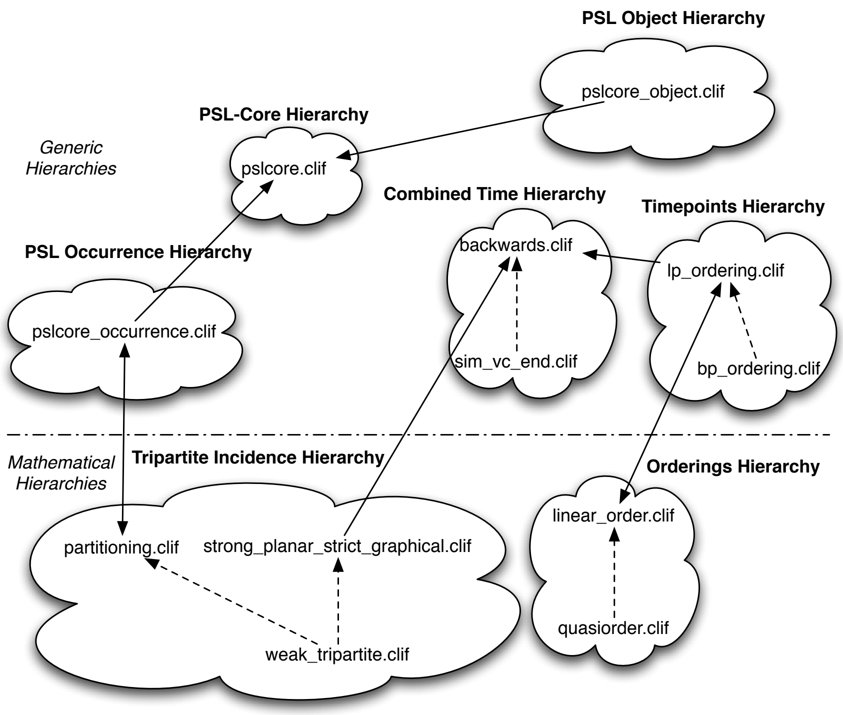

Figure 1. Two mathematical hierarchies (**Tripartite Incidence Hierarchy** and **Orderings Hierarchy**) from COLORE. All theories therein are knowledge patterns, three of them being interpreted by the generic ontologies `pslcore_occurrence.clif`, `backwards.clif`, and `lp_ordering.clif` that instantiate them with content-specific semantics. While the ontologies in the generic hierarchy **Combined Time Hierarchy** are reducible, the ones in **PSL Occurrence Hierarchy** and **Timepoints Hierarchy** are irreducible. Their root theories `pslcore_occurrence.clif` and `bp_ordering.clif` are thus candidates for formalized CPs. They are in turn used by other generic ontologies through composition (`pslcore.clif` and `pslcore_object.clif`) or extension (`backwards.clif`). *Figure generously contributed by Michael Gruninger.*

establishes a partial order over temporal objects, such as time points or intervals, is an ontological commitment, while stating that they are linearly ordered is an ontological choice because some theories of time may disagree with it. The ontological commitments of generic concepts or relations are theorems of the generic hierarchy's root theory. Mathematical hierarchies, on the other hand, are agnostic about ontological choices: what constitutes the root theory of a mathematical hierarchy is a matter of fiat definition.

This distinction between generic and mathematical hierarchies as one of intended semantics is outside the semantics expressible by first-order axioms. Thus, it is a distinction that must be made by humans, e.g. the maintainer of the repository. However, it is a binary choice that is easy for humans to make and does not overburden humans.

4.3. Root Modules of Generic Hierarchies Are Formalized CPs

The logical definitions and distinctions in ontology repositories together with the extralogical distinction between generic and mathematical hierarchies suffice to fully characterize a small subset of a repository's theories that are ideal candidates for formalized CPs. We will argue which theories these are and why they are more suitable than the other theories in the repository. To do so we specifically look at three criteria.

(i) Generic vs. mathematical theories Mathematical theories have no intended semantics and thus cannot capture the semantics of some real-world concepts and relations. This rules them out as candidates for any kind of content patterns, even though they are reusable modules. Instead, they closely align with *knowledge patterns* [1] as discussed in the next section. Generic theories, on the other hand, formalize a piece of generic content, may it be about time, space, processes, events or other kind of content. Thus, to mine the repository for formalized CPs, we must look in generic hierarchies.

(ii) Reducible vs. irreducible theories The repository contains both reducible and irreducible generic hierarchies. By definition, the reducible ones can be reduced to more general (and often smaller) theories, effectively breaking them into smaller modules; they are compositions of those smaller modules. While reducible theories may qualify as formalized CPs, the theories in irreducible generic hierarchies are more promising: they formalize pieces of content small enough such that they cannot be further broken down. One would expect those modules to be maximally reusable. In fact by definition all reducible theories already reuse some modules from irreducible hierarchies (similar to what has been called pattern composition in [14]), testifying to this reuse. Therefore, we should focus on finding formalized CPs among the irreducible generic theories.

(iii) Root vs. non-root theories Finally, because all hierarchies are closed under similarities, we can focus on the root theories rather than non-root theories as candidates for formalized CPs. The root theory of a hierarchy is its most general module that only contains the shared ontological commitments about the formalized set of concepts and relations while it abstracts away more restricted semantic interpretations. All non-root theories in the same hierarchy specialize this root theory (another form of reuse, compare [14]) by introducing additional ontological choices that are not deemed foundational for the described concepts and relations. Thus, instead of capturing a new pattern, a non-root theory merely specializes the more general pattern captured by the root theory it extends.

These three observations together identify the root theories of irreducible generic hierarchies as the most suitable candidates for formalized CPs. It remains to show that the root theories of generic hierarchies actually satisfy the three criteria for formalized CPs that conceptual CPs more or less lack (compare Sec. 3). They are (1) sufficiently formalized because all theories in the repository must be fully formalized in Common Logic [11] in order to properly place them into the COLORE repository; they are (2) sufficiently general because they capture the shared ontological assumption of a set of generic concepts and relations precisely because they are the root theory of that generic hierarchy; and they are (3) obtained by extracting what is common to existing ontologies about those concepts and relations, thereby capturing "best practice". Moreover, all nonconservative extensions within the same hierarchy testify to their root theories' reuse as does their inclusion in the reduction of reducible theories. Both kinds of reuse are well-documented through the logical relationships to other theories in the repository.

Thesis 1. *The root theory of every irreducible generic hierarchy in an ontology repository is a formalized CP.*

The so identified formalized CPs can be used in all the ways outlined in [14]. All six ways of their use are directly supported and documented through the following logical intertheory relationships present in COLORE.

- *import*: any extension—conservative or not—in COLORE, including language extensions that introduce additional primitive symbols, constitutes an import.

- *clone*: while so-called *shallow clones* are not supported by COLORE because such a clone would be logically identical to the original; so-called *deep clones* (more commonly called *instantiations*) are supported. Specifically, any theory synonymous[8] to an irreducible generic root theory in the repository constitutes a deep clone. Note, however, that the idea of logically synonymous theories is more general: it additionally encompasses logically equivalent theories that also differ in axioms but that can be proven logically equivalent after symbol renaming.
- *specialization*: any nonconservative extension within a hierarchy constitutes a specialization.
- *generalization*: generalization is a questionable operation for formalized CPs because if a formalized CP can be further generalized, it seems more appropriate to use the more general set of axioms for the pattern's formalization. Then only specialization is needed.
- *composition*: any extension across ontology hierarchies, i.e. any language extension, that extends at least two different irreducible generic root theories, constitutes a composition operation.
- *expansion*: any language extension constitutes an expansion operation.

Thus, the repository is readily able to support pattern-based ontology engineering, using the root theories of irreducible generic hierarchies as basic CPs.

Thesis 2. *Let T be the root theory of some irreducible generic hierarchy in an ontology repository. Then any reuse of T by another ontology O in the repository through either importation, deep cloning (instantiation), specialization, composition, or expansion is reflected in an extension or interpretation relationship between O and T in the repository.*

4.4. Theories of Mathematical Hierarchies are Knowledge Patterns

The term *knowledge patterns* [1] refers to an idea similar to that of formalized CPs. The term has been introduced independently of the notion of ontology design patterns, putting greater emphasis on the importance of reuse for "structurally similar patterns of axioms" through the definition of modules of axioms that formalize common classes of mathematical structures, thereby "explicitly modularizing and separating the abstract theories (the knowledge patterns) from the phenomena in the world which those theories are deemed to reflect" [1]. Reuse can be achieved by cloning patterns and appropriately renaming the symbols in the pattern's signature. For each instance where such a knowledge pattern is reused, the new ontology includes a logically equivalent subset of axioms. Such a new ontology is thereby reducible to a theory just describing the pattern plus additional theories (possibly containing more subtheories that are also reducible to patterns). Formally, each time a knowledge patterns is reused—"instantiated"—the new ontology *faithfully interprets* the theory defined by the knowledge pattern. Two examples are given in [1]: containers (with functions such as insertion and removal of entities) and different kinds of graphs.

Generic theories reuse mathematical structures, giving them more specific semantics and restricting the mathematical structure to fit the intended semantics. For the example of time theories brought up in [1], the underlying reused knowledge patterns are partial orders, whose involved ordered entities are assigned the semantics of time points or time

[8]Theories are synonymous iff they are logically equivalent with adequate translation definitions from the symbol of one to another, see [6].

intervals, and whose key order relations $<, >$ are assigned the meanings *before* and *after*. This demonstrates how knowledge patterns capture the *structure of the vocabulary*, thereby justifying the term *mathematical CP*. This is supplementary to *structural ODPs*, which focus on *logical/architectural structure of the entire ontology* and to *formalized CPs*, which focus on the formalized reusable content of generic concepts and relations.

An ontology repository consisting of generic and mathematical hierarchies and with a reduction order over hierarchies naturally reflects reuse of knowledge patterns: *all* theories in mathematical hierarchies can be treated as knowledge patterns in the sense of [1].

Thesis 3. *Every theory in a mathematical hierarchy in an ontology repository is a knowledge pattern.*

Not only the root theories of mathematical hierarchies, such as the most generic theory of partial orders or the most generic theory of directed graphs, qualify as knowledge patterns, but all theories of partial orders or of graphs qualify. This is demonstrated by the example of graphs given in [1]: the theory T_{dag} (directed acyclic graphs) and its restriction to the theory $T_{\mathrm{dag-blockable}}$ (blockable directed acyclic graphs) are treated as knowledge patterns. Both utilize the same primitive signature $node(x)$, $to(x, y)$, and $reaches(x, y)$, which suffice to define the concept $blocked(x)$ (and $unblocked(x)$). Thus, both are in a single hierarchy and $T_{\mathrm{dag-blockable}}$ is a nonconservative extension of T_{dag} and therefore definitely not a root theory, but still a knowledge pattern[9].

Reuse of knowledge patterns is either through instantiation or expansion. Either one manifests itself in a relative interpretation relation between a (generic or domain) ontology and a mathematical theory in the repository.

Thesis 4. *Let O and T be two distinct ontologies in an ontology repository, with T being contained in a mathematical hierarchy. Then O instantiates or expands T if, and only if, O is not contained in a mathematical hierarchy and O relatively interprets T.*

5. Discussion

We expressed two central kinds of ontology design patterns, namely *Formalized Content Design Patterns* (formalized CPs) and *Knowledge Patterns*, in terms of logically well-defined concepts and intertheory relationships in ontology repositories, and thereby demonstrated that the COLORE repository already contains hundreds of knowledge patterns and dozens of formalized CPs[10]. Because many knowledge patterns are already instantiated by non-mathematical ontologies (generic and domain- or application-specific ontologies) residing in the repository, they have taken initial hurdles for passing the "best practice" test. Equally, formalized CPs can been identified in the repository through similarities between multiple ontologies, which ensures that they are "best practices" as well. By mapping these two kinds of ontology patterns to logic-based criteria, we have provided an objective basis for the identification of suitable patterns of either kind. These patterns can be easily extracted from the repository for reuse independent of the reposi-

[9]The hierarchy that contains both theories is different from the `graph` hierarchy, `http://colore.oor.net/graphs/`, available in COLORE that utilizes only a single primitive relation of adjacency and cannot define reachability and blocking.

[10]The mapping between patterns and ontology modules extends to other repositories that include and relate ontologies formalized in different logical languages. For example, ontological repositories that explicitly express intertheory relations in a a higher-order logical language such as the Distributed Ontology Language (DOL) outlined in [13] are equally suited to identify and document ontology patterns.

tory.Therefore, it may make more sense to extend existing ontology hierarchies and define new ones instead of developing CPs from scratch. The repository serves as an objective way of ensuring that only the most suitable theories become patterns. Furthermore, the logic grounding enables using the repository for pattern-based ontology design.

A recent small survey of ontology pattern use [17] identified the lack of detailed documentation and, specifically, the lack of abundant examples as a major impediment for use and adaptation of patterns. By identifying mathematical and generic patterns in a repository, this issue can be easily and permanently addressed: each pattern is linked to ontologies that use the pattern through logical relationships such as extensions or interpretations. Therefore, one can pick out a pattern and follow all logical relationships (which are explicitly stored in the repository) to specific examples of their use. This applies both to mathematical and generic ontology patterns. Adding more logical relationships immediately makes more examples of pattern reuse visible. Thereby, the use of ontology repositories helps realize the original ideas for the role of CPs from [1, 4] that envisioned constructing hierarchies of patterns through importation, instantiation (cloning), specialization, generalization, composition, and expansion.

References

[1] Clark, P., Thompson, J., Porter, B.: Knowledge patterns. In Staab, S., Studer, R., eds.: Handbook of Ontologies. Springer (2003) 191–207

[2] Falbo, R.A., Guizzardi, G., Gangemi, A., Presutti, V.: Ontology patterns: Clarifying concepts and terminology. In: Workshop on Ontology and Semantic Web Patterns (WOP 2013). (2013)

[3] Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley (1995)

[4] Gangemi, A.: Ontology design patterns for semantic web content. In: Int. Semantic Web Conf (ISWC-2005). (2005)

[5] Grüninger, M.: Verification of the OWL-Time ontology. Int. Semantic Web Conf (ISWC-2011). (2011)

[6] Grüninger, M., Chui, C., Hahmann, T., Katsumi, M.: A sideways look at upper ontologies. In: Conf. on Formal Ontology in Inf. Systems (FOIS-14). (2014 (to appear))

[7] Grüninger, M., Fox, M.S.: The role of competency questions in enterprise engineering. In: IFIP WG5.7 Workshop on Benchmarking – Theory and Practice, Trondheim, Norway. (1994)

[8] Grüninger, M., Hahmann, T., Hashemi, A., Ong, D., Ozgovde, A.: Modular first-order ontologies via repositories. Applied Ontology **7**(2) (2012) 169–209

[9] Hahmann, T.: A Reconciliation of Logical Representations of Space: from Multidimensional Mereotopology to Geometry. PhD thesis, Univ. of Toronto, Department of Comp. Science (2013)

[10] Hu, Y., Janowicz, K., Carral, D., Scheider, S., Kuhn, W., Berg-Cross, G., Hitzler, P., Dean, M., Kolas, D.: A geo-ontology design pattern for semantic trajectories. In: Conf. on Spatial Inf. Theory (COSIT-13), Springer (2013)

[11] Intern. Electrotechnical Commission (ISO/IEC) 24707: Common Logic (CL). http://standards.iso.org/ittf/PubliclyAvailableStandards/c039175_ISO_IEC_24707_2007(E).zip (2007)

[12] Katsumi, M., Grüninger, M.: Specifying ontology design patterns. In: 4th Workshop on Ontology and Semantic Web Patterns (WOP2013). (2013)

[13] Mossakowski, T., Lange, C., Kutz, O.: Three semantics for the core of the distributed ontology language. In: Conf. on Formal Ontology in Inf. Systems (FOIS-12), IOS Press (2012)

[14] Presutti, V., Gangemi, A.: Content ontology design patterns as practical building blocks for web ontologies. In: Conceptual Modeling (ER 2008). LNCS 5231, Springer (2008) 128–141

[15] Scherp, A., Saathoff, C., Franz, T., Staab, S.: Designing core ontologies. Applied Ontology **6**(3) (2011) 177–221

[16] Sinha, G., Mark, D., Kolas, D., Varanka, D., Romero, B.E., Feng, C.C., Usery, L.E., Liebermann, J., Sorokine, A.: An ontology design pattern for surface water features. In: 8th Int. Conf. on Geographic Information Science (GIScience 2014). (2014 (to appear))

[17] Warren, P.: Ontology patterns: a survey into their use. Technical Report kmi-14-02, Knowledge Media Institute (2014)