An Enhanced Graph-Oriented Approach for Change Management in Distributed Biomedical Ontologies and Linked Data

Arash Shaban-Nejad
McGill Clinical & Health Informatics, Departments of Epidemiology & Biostatistics, Faculty of Medicine
McGill University
Montreal, Quebec, CANADA
arash.shaban-nejad@mcgill.ca

Volker Haarslev
Department of Computer Science and Software Engineering
Concordia University
Montreal, Quebec, CANADA
haarslev@cs.concordia.ca

Abstract— **This paper reports the summary and results of our research on providing a graph oriented formalism to represent, analyze and validate the evolution of bio-ontologies, with emphasis on the FungalWeb Ontology. In this approach Category theory along with rule-based hierarchical distributed (HD) graph transformation have been employed to propose a more specific semantics for analyzing ontological changes and transformations between different versions of an ontology, as well as tracking the effects of a change in different levels of abstractions.**

Keywords-component; bio-ontologies, ontology evolution, category theory, graph transformation, fungal genomics

I. Introduction

Ontologies are widely used as a vehicle for knowledge management in current biomedical applications. The main components of ontologies are concepts, relations, individuals (instances) and axioms. Ontologies are evolving over time in order to fix errors, reclassify the taxonomy, add/remove concepts, attributes, relations and instances. One of the most challenging issues in ontology change management is keeping ontologies consistent when changes occur. Generally most of the existing change management approaches have been faced with the following three issues:

1- Overreliance on human factor;
2- Lack of a suitable formalism to deal with temporal abstract notions;
3- Neglecting the interactions with other existing ontologies and focusing on changes in internal structure of ontologies.

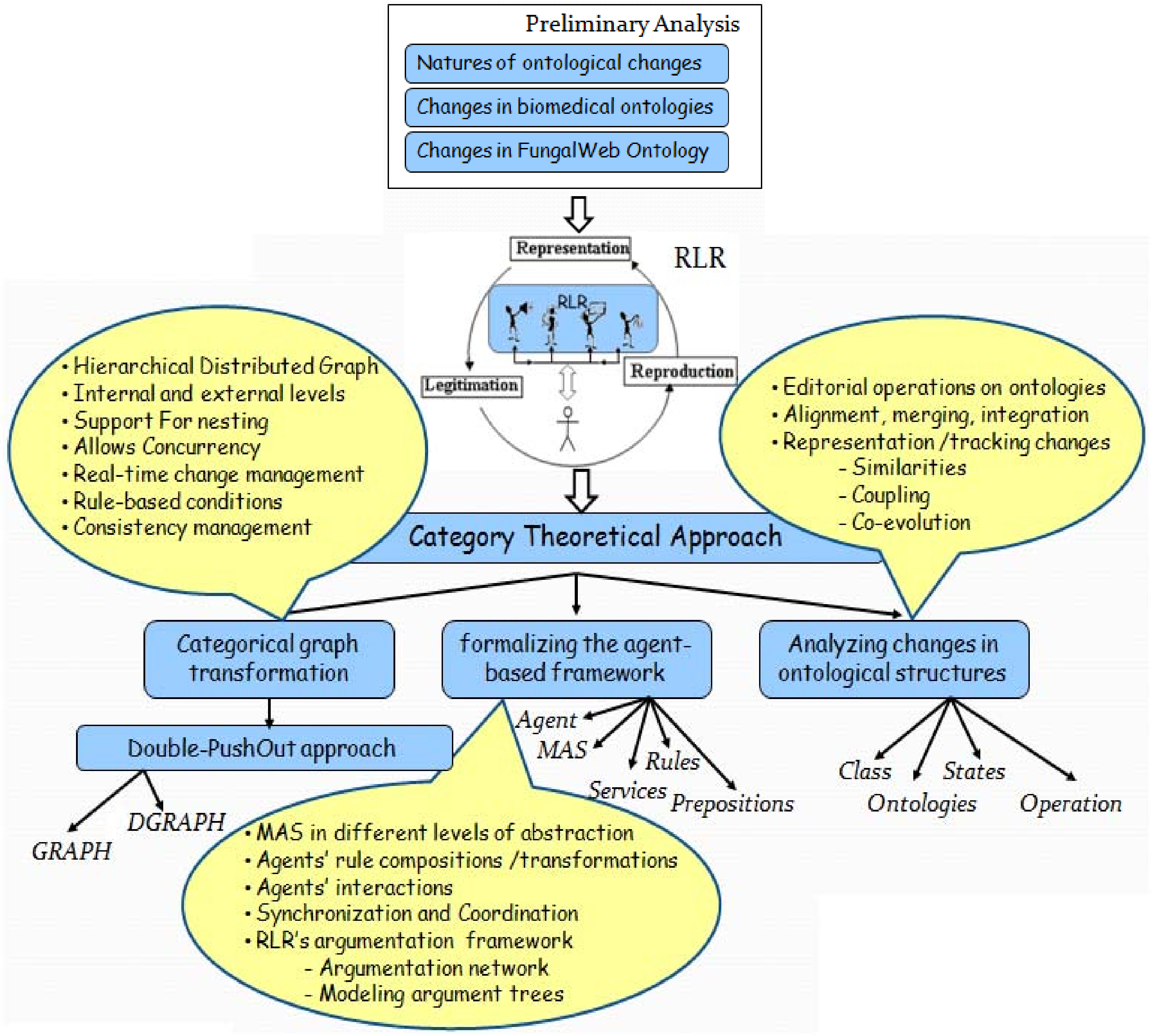
In order to address these issues our study has been focused on finding a suitable formalism to capture, represent and analyze the ontological alterations in the domain of life sciences with minimum human intervention. Our proposed formalism can overcome the *decidability* issue that occurs in temporal description logics and the rigidity of OWL's single semantic structure. Moreover we specifically analyze how different changes can affect the dependent artifacts in a distributed environment. To this end, after analyzing the context of the problem and reviewing other existing techniques for change management in some existing biomedical ontologies [1], we presented a novel multi-agent-based approach, RLR (Represent, Legitimate, and Reproduce) [2, 3] to manage the evolving structure of biomedical ontologies in a semi-automatic and consistent manner with reproducible results. The RLR framework aims to assist and guide ontology engineers through the change management process in general, and aids in tracking and representing the changes, particularly through the use of graph transformation empowered with category theory as a mathematical notation, which is independent of any specific choice of ontology language or particular implementation. As we will demonstrate throughout this manuscript categories and graph transformation – more specifically hierarchically distributed (HD) graph transformation– provide abstract yet expressive enough formalism to address the second and third issues stated above. HD graph transformation as an adapted type of standard graph transformation has been employed to maintain the hierarchically structured knowledge in the Semantic Web environment. The graph transformation rules describe the structural changes placed during a knowledge base operation. To perform the transformation we employ the well-known category theoretical method double-pushout [4]. In contrast to some of the existing works on ontology evolution, we focus on changes in distributed ontologies, not as standalone artifacts but in contact with other ontologies in an open semantic web environment. We demonstrate the technical correctness and feasibility of our approach through a set of case studies. Figure 1 demonstrates a general overview of RLR and the associated categorical graph-oriented formalism for managing changes in biomedical ontologies.

II. Changes in Biomedical Ontologies

There are currently a growing number of ontologies and controlled vocabularies in various areas of life sciences. Several ontologies in the biomedical domain are known to be seriously defective in both terminological and ontological perspectives. In our research we selected some of the most popular ontologies and controlled vocabularies in health science to find the evidences for various types of possible changes. The following ontologies have been selected based on several criteria such as availability, popularity, complexity and accessibility to the source and documentation: The Gene Ontology (GO) [5], Clinical Term Version 3 (The Read Codes) [6], Health Level 7 (HL7) [7], UMLS Semantic Network [78], The Foundational Model of

Anatomy (FMA) [8], Medical Subject Heading (MeSH) [9] and Terminologia Anatomica (TA) [10].

Based on our research of the literature, observing different releases of bio-ontologies, surveys, and interviews with several domain experts and ontology engineers, we distinguished several changes and among them we selected about 74 different types of changes that frequently occur in life cycles of existing biomedical ontologies. These changes are classified under ten general terms: addition, deletion, retirement (obsoletion), merging, splitting, replacement (edit), movement, importing, integration, or changes to file structure.



Preliminary Analysis
Natures of ontological changes
Changes in biomedical ontologies
Changes in FungalWeb Ontology
Representation
RLR
RLR
Legitimation
Reproduction
• Hierarchical Distributed Graph
• Internal and external levels
• Support For nesting
• Allows Concurrency
• Real-time change management
• Rule-based conditions
• Consistency management
• Editorial operations on ontologies
• Alignment, merging, integration
• Representation /tracking changes
- Similarities
- Coupling
- Co-evolution
Category Theoretical Approach
Categorical graph transformation
formalizing the agent-based framework
Analyzing changes in ontological structures
Double-PushOut approach
GRAPH
DGRAPH
Agent
MAS
Rules
Services
Prepositions
Class
States
Ontologies
Operation
• MAS in different levels of abstraction
• Agents' rule compositions /transformations
• Agents' interactions
• Synchronization and Coordination
• RLR's argumentation framework
- Argumentation network
- Modeling argument trees

Figure 1. A general representation of RLR and the associated categorical graph-oriented formalism for managing changes in biomedical ontologies. As shown category theoritical approach has been used to perform change management in bio-ontologies by defining categories of *Class,States, Ontologies and Opearation* to analyze ontological changes at both internal and external levels. Also categories of *Agent, MAS,Services, Rules* and *Preposition* have been used to formalize agents interactions and negotiation [28]. Moreover categorical representation has been employed as a main formal method for performing hierarchical distributed graph transformation, specifically through two categories of GRAPH and DGRAPH (distributed graphs).

For the application scenario we consider changes in the FungalWeb Ontology [11], which is an integrated formal bio-ontology in the domain of fungal genomics, and consists of several interrelated ontologies and data sources (Figure 2). The Fungal taxonomy is not stable. Most of the alterations are changes in names and taxonomic structure and relationships. Fungal names reflect data about the organisms; thus, as our understanding of the relationships among taxa improves, these names will need to be changed, as they will no longer convey the correct information to the user. Most fungi names are currently based on phenotypes. These name changes may cause confusion and affect the validity of different queries. The morphological conceptualization of fungi is not sufficient, and will no longer work because all of the names based only on morphology must be re-evaluated. Also, the phylogenetic-based conceptualization has its own limitations, since the decision of where to draw the line between different species is not always easy to make [12].

Moreover many terms in current medical mycology vocabularies describing skin disorders originate as verbal descriptions of appearance, foods, people, mythological and religious texts, geographical places, and acronyms [13]. Many names and terms are highly dependent on individual or regional preferences, causing redundancy, vagueness, and misclassification in current vocabularies. Thus, we study various alterations in both fungal taxonomy and fungal disease classification. As an example of changes in fungal terminologies, one can see several changes in the name of pathogenic fungi *Trichophyton* family (i.e. *Trichophyton Soudanense*, *Trichophyton megninii,* and *Trichophyton equinum*) in relatively short period of time. As another example, the pathogenic fungus *Candida glabrata* is now called *Torulopsis glabrata* [14].

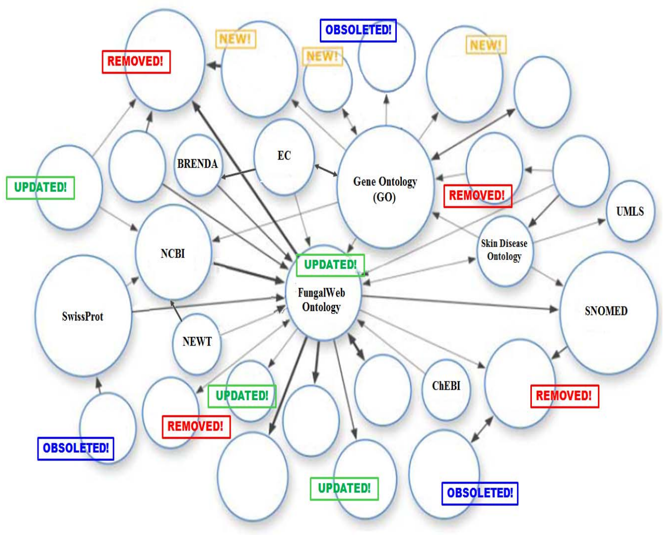


Figure 2. Interrelated distributed ontologies, knowledge bases and data sources in the FungalWeb infrastructure.

Usually changes in fungi taxonomy alter the related disease name and description. For instance, the name of the fungus, *Allescheria boydii* which can cause various infections in humans, was changed to *Petriellidium boydii* and then to *Pseudallescheria boydii* within a short time [15]. Consequently, the infections caused by this organism were referred to as *allescheriasis*, *allescheriosis*, *petriellidosis*, and *pseudallescheriosis* in the medical literature [16].

To manage this process of continuous change, we rely on ontological conceptualization, where names in a taxonomy are only meaningful once linked to descriptive datasets, which are extracted and managed from various databases and literature in an integrated environment.

III. The RLR Framework

The RLR framework for change management consists of a set of intelligent agents designed to perform several intelligent tasks including learning, reasoning, capturing changes and negotiation within a collaborative environment. In a typical scenario within the RLR argumentative architecture, a user (human or agent) initially sends a request to an ontology engineer for a particular change in the ontological structure. Based on the system's background knowledge and the choice of the ontology engineer, various options are available to implement a change. A negotiation agent, along with a reasoning agent, provides arguments for the acceptance or rejection of a change proposal. Also an "Argument Generator" determines appropriate responses based on the negotiation rules. Different arguments attack each other to enforce their rules and defeat their peers by sending counter-arguments. The inferred arguments can increase the possibility of higher quality agreements [17]. The Negotiation Protocols in the RLR architecture contain the rules that dictate a protocol. As the knowledge base is used and evolves, the historical information about different changes will be accumulated in the change logs. This information will be used by the learner agent, which acts as a basis for a recommender system, to propose different alternatives for the implementation of future changes. The reasoning and negotiation agents can change the rules if necessary and send modifications to the learning agent. In order to maintain agents' argumentation for automation of ontology evolution, we employ the "dialectical databases" [17]. In argumentation-based multi-agent systems, a dialectical database tends to improve the speed of inference responses by storing pre-compiled knowledge about potential dialectical trees [18]. The dialectical trees represent sets of possible dialectical confrontations between the arguments to accept or deny a proposal to deal with a particular change [19]. For the detail on the structure of RLR we refer the reader to [2, 3].

We use category theory and graph transformation to explore systematic changes in ontologies, analyze rule based transformations, and study various dependencies between the ontological elements, as well as formalizing agents' interactions and communications in the RLR framework.

IV. Category Theory and Graph transformation

Category theory is a relatively new domain of mathematics, introduced and formulated in 1945 [20]. Category theory is closely connected with computation and logic, which allows an ontology engineer to implement different states of design models to represent the reality. Using categories, one can recognize certain regularities to distinguish a variety of objects, capture and compose their interactions and differentiate equivalent interactions, identify patterns of interacting objects and extract some invariants in their action, or decompose a complex object into basic components. Categorical notations consist of diagrams with arrows. Each arrow $f: X \rightarrow Y$ represents a function. A Category C includes: A class of objects and a class of morphisms ("arrows"), and for each morphism f

there exists one object (A) as the domain of *f*, and one object (B) as the codomain; For each object A, an identity morphism, which has domain A and codomain A ("ID_A"); and For each pair of morphisms *f*:A→B and *g*:B→C, (i.e., cod(*f*) = dom(*g*)), a *composite morphism*, *g o f*: A→C exists. Some of the primitive constructors of category theory that we use in our framework for ontology change management are as follows: Products, Co-products, Functors, Natural Transformation, Pushout and Pullback. More information on these categorical notions can be found in [21]. For example in the Fungalweb ontology Fungal Meningitis is an infectious disease caused by a fungus, e.g. *Cryptococcus Neoformans, which* is typically seen in patients with immune deficiency such as AIDS. It usually results from an infection that spreads to a patient's brain from another part of her body. This disease has been a subject for study in both dermatology and neurology for a long time. The knowledge about this disease (i.e. symptoms, causes, etc.) are scattered in several existing ontologies and knowledge bases, which need to be aligned. We model the alignment of two ontologies by means of a pair of ontology mappings from a bridge ontology using categorical notations (Figure 3).

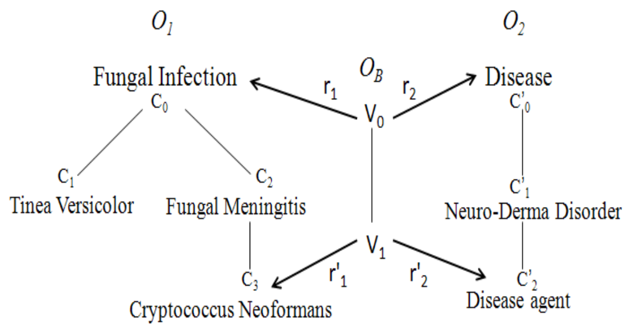


Figure 3. The categorical representation of the alignment between two ontologies O_1 (Fungal disorders), and O_2 (Diseases) using a bridge ontology O_B and a set of bridge axioms (r) .

In order to achieve a composite knowledge of the disease's properties we have used the categorical product to represent this integrated view (Figure 4). As can be seen in the Figure 4 medical specialty is the product arrow of the two branches in medicine, which includes the attributes of both domains. In order to merge two unrelated ontologies we can simply perform the disjoint union (or co-product).

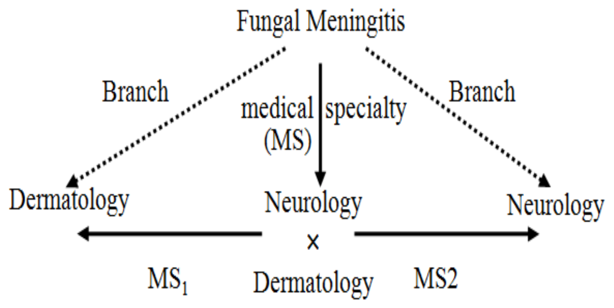


Figure 4. Determining the medical specialty for a particular disease through product.

The rule-based graph transformation can be studied based on the following three activities [22]:

- Creating the conceptual generalizations of the reality and transferring them from "reality" to its representation in a model;
- The definition of rules as specifications of state transformations;
- Using graphs as a means to represent snapshots, concepts, and rules.

Generally applying a transformation rule (production) p: L → R denotes finding a proper match of L (Left hand side) in the source graph and replacing L by R (Right hand side), leading to the target graph of the graph transformation. Following the double-pushout approach (DPO) [4], a transformation rule is defined [23] as a pair $t: L \leftarrow I \rightarrow R$ of morphisms $l: I \rightarrow L$ and $r: I \rightarrow R$ such that l is injective, where the graphs L and R are called the left and right-hand sides respectively, and I is called the interface or gluing graph. It is not necessary for the morphism $r: I \rightarrow R$ to be injective, which allows one to identify different nodes or edges in various transformations. Also, the injectivity of $l: I \rightarrow L$ ensures the uniqueness of the results in backward tracing in a transformation. The rule t transforms a graph O_G into a graph O_H, denoted by $O_G \Rightarrow_t O_H$ if there is an injective occurrence morphism $m: L \rightarrow O_G$, and two pushouts as represented bellow:

$$\begin{tikzcd} L \arrow[d, "m"'] & I \arrow[l, "l"'] \arrow[r, "r"] \arrow[d] \arrow[l, phantom, shift right=8, "(I)"] \arrow[r, phantom, shift right=8, "(II)"] & R \arrow[d] \\ O_G & K \arrow[l] \arrow[r] & O_H \end{tikzcd}$$

Figure 5. Double-Pushout approach for graph transformation

In summary DPO should be performed through the following steps when a rule $L \leftarrow I \rightarrow R$ is given:

(i) Find the elements of L in the given graph O_G, i.e. a match m: L → O_G; *(ii)* Delete from O_G all the elements specified in L, which are not in the gluing graph *I*. This means to find a graph K and graph morphisms K → O_G, and *I* → K such that the square is a pushout; (*iii*) Add to graph K all the elements of R, which are not in the gluing graph *I* and create the second pushout and obtain a derived graph O_H.

V. Employing HD Graph Transformation for Ontology Change Management

Changes in an area due to technical, industrial, cultural, or social matters force the existing systems and applications to adapt themselves to the new state. Particularly, large systems and knowledge bases built upon smaller reusable sub-systems are in greater danger and should be continuously monitored to ensure the correctness and consistency of the entire infrastructure. In an ontological sense, concepts in an ontology naturally match with nodes of a graph, while the relationships in an ontology correspond to edges. The graph-based representation of the biomedical ontologies has a great tendency to become large, complex, and hard to grasp, understand, or maintain in a very short time. In applications dealing with compound graphs in layered organizations, the notion of graph can be extended to hierarchical graph. In order to mimic the actual nested hierarchical structure of the Semantic Web, where information is distributed in the nodes (graphs) and edges (relations between the graphs), we employ hierarchical distributed (HD) graphs [24], which enables us to perform the transformation on different levels of abstraction. The hierarchical graphs have richer semantics and are more expressive in comparison with regular flat graphs. In addition, they reduce the complexity of representation of large interrelated systems by allowing one to describe a system on a more abstract level through hiding the irrelevant details in encapsulated sub-graphs [25]. Hierarchical graph transformation can be performed using the extended double-pushout notion to represent various aspects of dynamic structures (e.g., the rearrangements of some temporal parts, describing the changes in relations, creation/deletion of communication channels, and performing operations such as "splitting" a graph into two or more graphs or "joining" distributed graphs into one graph). As defined by [26] distributed graphs distinguish between two levels, namely local (internal), and network (external or lattice) (Figure 6). The communication between internal graphs can be performed via interfaces.

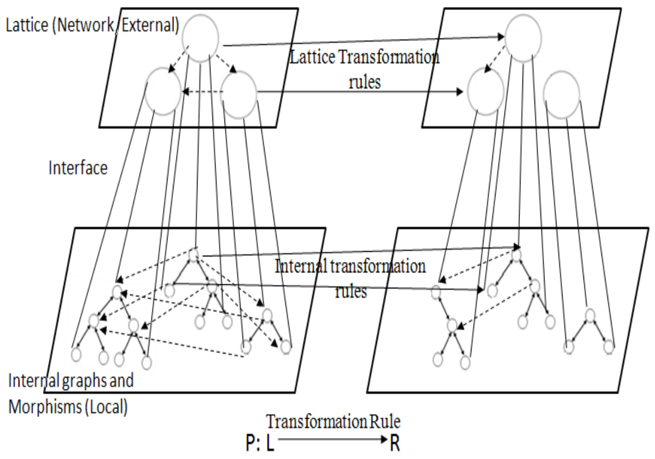

Lattice (Network/External)
Lattice Transformation rules
Interface
Internal transformation rules
Internal graphs and Morphisms (Local)
Transformation Rule
P: L → R

Figure 6. Representation of a distributed graph transformation along with the distributed transformation rule, which regulates the transformation of different interconnected ontologies in two abstraction levels, namely internal and lattice.

In our model, the hierarchical graph (the lattice) consists of a set of internal graphs (which may be hierarchical graphs as well), the root of the hierarchy, and a set of edges that relates the internal graphs to each other.

Each editorial action is expressed through a graph transformation and every state of the ontological structure is modeled in a graph with the nodes denoting objects and the edges representing the connections linking them. The categorical graph grammar [27] supports the flexible change of complex interrelated compositions while providing explanations for corresponding actions performed by graph transformation. Various states can be produced by internal or external actions, and their communications can be modeled and simulated using graphs and state transitions, then represented and described by means of graph transformation. The double-pushout technique has been extended from flat to hierarchical graphs [23], where the associated transformation rules can be applied at all hierarchical levels. This facilitates changes of the graph's entries (i.e., by insertion or deletion) regardless of their size and configuration, with adaptation of the "dangling condition" from the flat graphs transformations. The compound state of the entire system can be known by analyzing several other internal graphs, each having an internal state and behavior. There are also lattice-like dependency graphs representing dependencies between different internal graphs. In the process of change management for the lattice-like structure, several concerns related to sequential, parallel, or concurrent evolution of its components arise.

Different ontologies in Semantic Web are usually connected in a lattice-like structure and interact with each other through one or more interfaces. This lattice can be modeled as a directed graph with individual ontologies (internal graphs) as its nodes and the links between these ontologies as its edges. The described configuration is analogous to HD-graph [24, 29], where each of the links connecting the internal graphs contains a graph morphism specifying the relation between two internal states. When the internal graphs are faced with any change (e.g., adding/deleting a concept or relation), their state would be changed, which would affect other dependent graphs, and a synchronization unit within the RLR framework, which stores all the states in the change logs, forces the lattice-like structure and the mediator interface to change their states accordingly. This structure can be modeled in two different but related planes, namely conceptual (shows all existing and potential relations, paths, and their revisions) and operational (shows only actual existing nodes and relations).

In order to categorically analyze the distributed transformations we define the category of distributed graphs DGRAPH with distributed graphs as objects and distributed graph morphisms as arrows. Then we can define a transformation using double-pushout, in the category DGRAPH, while distributed graphs and distributed graph morphisms serve as objects and arrows for the category DGRAPH [24]. For the details of proofs and other related categorical notions in distributed graph transformation one may refer to [24, 30].

In summary we have applied our combined category-graph oriented method for:

A. Analyzing Events and Actions in Rule-Based Model Transformation

In order to analyze different events that trigger actions during the ontological evolution process, we consider events as part of the rule condition in a graph transformation. The actions are described by productions and the events will occur if the gluing condition [31] holds and certain predefined conditions (e.g. ontological rules and axioms) are assessed to be true. We also use hierarchical distributed graph rules covering both internal and external production describing the internal and external actions respectively. A transformation can be defined to be conditional [32] in such a way that under certain conditions, the graph production (rules) transform a source graph into the target graph. These conditions, which impose a set of restrictions on the transformation processes, can help one to avoid inconsistencies and conflicts (e.g., the conflicts due to dangling edges). A transformation rule determines conditions such as: 'the deletion of a lattice node should be performed after deleting its corresponding internal graphs' (see Figure 7). As long as the actions (e.g. deletion, insertion) do not violate the defined conditions in the production rules several actions can be executed in parallel at the local level (e.g deletion/creation of internal elements). As mentioned, the external lattice production describes the structural changes of the external graph, and we can model the external actions using a hierarchical distributed graph production in such a way that an identical production for the internal graphs of every node of the external graph (individual ontological structures) must be performed. If the stated predefined conditions for insertion/deletion of the nodes in the internal graphs are satisfied, then the hierarchical distributed graph production can be applied at the external (lattice) level (for adding/deleting edges, a set of morphisms will be described instead).

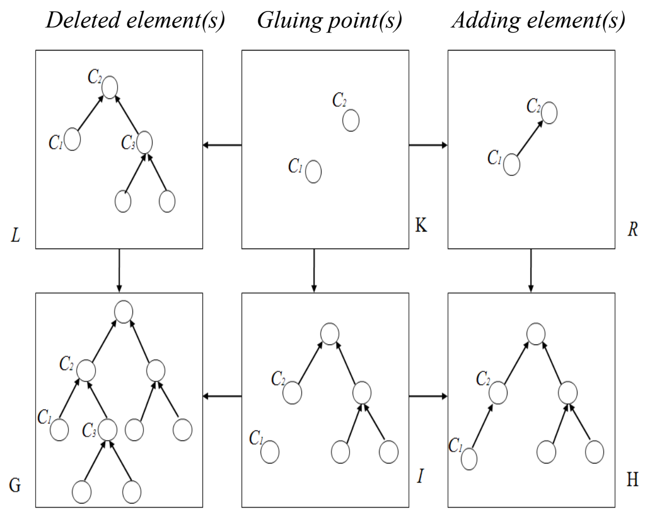


Figure 7. A rule-based transformation that indicates the changes of the partial taxonomy G to H based on the defined rule "delete a parent node". The upper part represents the transformation rule, and the bottom left shows a given graph and the bottom right demonstrate the result of the transformation, which has been obtained by following the three steps in DPO.

The rule conditions, which impose a set of restrictions on the transformation processes, can help one to avoid inconsistencies and conflicts (e.g., the conflicts due to dangling edges).

B. *Synchronization and Coordination*

The hierarchical distributed graph productions are used for synchronization purposes by checking whether the external production is identical (or compatible) with what is performed by internal actions. More precisely, it checks if the lattice nodes and edges, in coordination with internal actions, have been identically replaced in the interface with respect to the gluing condition. For example, a graph production can describe a synchronous communication channel [29] between two different versions of an internal graph by highlighting the revisions in the original state and the current state through the use of an interface graph. Later on, the action that causes a change in the internal graph needs to be synchronized with other actions on dependent internal graphs and finally with the actions that alter the external graph. In real world applications, this synchronization usually results in a series of mappings between the previous and current states.

C. *Rule-based Patterns for Transformations*

After each change, the system needs to be verified for consistency. In order to preserve the ontological elements' identities and guarantee the consistency and integrity of the changes, we define a set of pre- and post-conditions to be satisfied. To hide unnecessary details, the change processes and related interactions are performed via interfaces. By using category theory for defining the patterns, we focus on the interactions between objects rather than their internal structure. In summary, in our categorical representation of a hierarchical graph organization, anything other than nodes and edges (e.g., attributes such as data type properties for ontologies) are supposed to be marginal and not essential. Thus, the notion of graph transformation can be defined [33] as $G,R \Rightarrow C,E$, with G, R, C, E respectively indicating a category of graphs, a category of rules, a category of control conditions, and a category of graph expressions. Modeling the notion of graph transformation in an abstract way is significant in the sense that it hides the marginal information, which does not explicitly contribute in the transformation process.

D. *Similarity Checking and Traceability*

We applied the graph comparison methodology presented in [23] along with categorical isomorphisms to compare the contents of two graphs by considering the number of nodes and edges. The comparison has been performed based on applying the rules while considering the hierarchical dangling condition, to check whether a specific sub-graph exists or not.

VI. EVALUATION A ND CONSISTENCY MANAGEMENT

The consistency management in our model includes several options:

a) Enforcing the actions for prohibiting the alterations that may lead to inconsistencies that often inherit to different versions and endure over the substantial part of the ontology's life cycle. This has been done by defining a set of conditions on transformations.

b) Employing tools such as AGG [34] for automatically checking the consistency of a transformation.

c) Isomorphic Reasoning and Commutative Inference: In order to validate the categorical diagrams the partial isomorphism in the semantic web environment is defined based on the similarity in structural relationships between syntax, semantics, and the resources of the knowledge in ontological frameworks.

From a categorical point of view, the simplest type of isomorphic reasoning involves an explicit and continuous mapping of the correspondences and similarities at the syntactic level while ignoring the semantics. This method enables us to perform reasoning about the dynamic structure of ontologies. For example, in the case of context change in ontology evolution, since the applicability of specific knowledge in one context does not automatically indicate the validity of the reasoning in the new context, thus the isomorphism between different states of the ontological structures and the knowledge they implied needs to be carefully analyzed. A common sense approach to get insight into a categorical diagrammatic structure and trace its various states, is to follow and chase the diagrams depicting the objects and morphisms, to check whether the diagram is commutative or not and ensure the equality of the compositions. A diagram is commutative "if and only if whenever p and p' are paths with the same source and target, then the compositions of morphisms along these two paths are equal" [35]. Putting two commutative diagrams together yields another commutative diagram. The diagram chasing along with commutative inference allow us the state space analysis to examine all the potential state transitions based on a derived transformational pattern. Therefore, one of the fundamental functionalities in ontology engineering that is the traceability of isomorphic reasoning processes through time from an initial ontology version to its current operational version can be performed.

VII. DISCUSSION

In the employed graph transformation approach, we restricted ourselves to using typed labeled graphs; however, in order to increase the expressivity of the graph representation, one may want to employ hypergraphs instead. Although using hypergraphs increases the expressivity of our formalism, it also induces a tremendous amount of complexity on the reasoning process (comparable with using OWL Full as the representation language).

Based on our experiments dealing with category theory, we feel that this formalism still has plenty of potential left to be used for ontology change management; thus, the categorical constructors such as *sketches*, *n-categories,* and *enriched categories* are due for examination in future work.

From our experience so far, some of the advantages of our introduced model are:

(i) The representation of events, time, actions, and operations employed in different scenarios of a dynamic ontological framework is an effective way to trace model changes;

(ii) The independency of the framework from any particular domain, algorithm, protocol, or implementation language and its abstractness makes it more flexible for reuse in many application domains that use different formalisms and platforms;

(iii) Employing transformation rules to perform changes ensures the consistency of the evolving ontologies in different states;

(iv) Following the double-pushout approach (DPO) for defining model transformation, which isolates the parts that remain unchanged, enables concurrent changes within an integrated knowledge-based system with minimum interruption to the system's operation;

(v) The abstract categorical notions and their ability to specify objects and their relations in different levels of granularities, together with graph oriented semantics, enable us to describe the complex evolving structure in a consistent manner, which is beyond the capability offered by OWL's single semantic structure.

REFERENCES

[1] A. Shaban-Nejad, and V. Haarslev, "Bio-medical Ontologies Maintenance and Change Management." In: Biomedical Data and Applications. Studies in Comput. Intelligence, vol. 224, Springer, 2009, pp.143-168, doi: 10.1007/978-3-642-02193-0_6.

[2] A. Shaban-Nejad, and V. Haarslev, "Incremental biomedical ontology change management through learning agents," Proc. of the 2nd KES International Symposium on Agent and Multi-Agent Systems: Technologies and Applications (KES-AMSTA 08), Springer, LNCS 4953, 2008, pp. 526–535, doi: 10.1007/978-3-540-78582-8_53.

[3] A. Shaban-Nejad, M. Kassab, O. Ormandjieva, and V. Haarslev, "Managing Requirements Volatility in an Ontology-Driven Clinical LIMS Using Category Theory," Int'l Journal of Telemedicine and Applications 2009, Article ID917826, 14 pages, doi:10.1155/2009/917826.

[4] H. Ehrig H, M. Pfender, and H. J. Schneider, "Graph grammars: An algebraic approach," Proc. of 14th Symposium on Foundations of Computer Science (FOCS 73), IEEE press, 1973, pp. 167-180, doi: 10.1109/SWAT.1973.11.

[5] M. Ashburner, C. A. Ball, J.A. Blake, D. Botstein, H. Butler, J. M. Cherry, A. P. Davis, K. Dolinski, S. S. Dwight, J. T. Eppig, M. A. Harris, D. P. Hill, L. Issel-Tarver, A. Kasarskis, S. Lewis, J. C. Matese, J. E. Richardson, M. Ringwald, G. M. Rubin, and G. Sherlock, "Gene Ontology: tool for theunification of biology," Nat Genet., vol. 25, 2000, pp. 25-29, doi: 10.1038/75556.

[6] NHS Information Authority: The Clinical Terms Version 3 (The Read Codes): Managing Change: Description Change File. Ref # 1999-IA-173 v1.0 issue, March 2000.

[7] Health Level 7 Reference Information Model
http://healthinfo.med.dal.ca/hl7intro/CDA_R2_normativewebedition/infrastructure/rim/rim.htm

[8] C. Rosse, and J. L. Mejino Jr, "A reference ontology for bioinformatics: the Foundational Model of Anatomy," Journal

Biomed Inform. vol. 36(6), 2003, pp. 478-500, doi:10.1016/j.jbi.2003.11.007.

[9] M. H. Coletti, and H. L. Bleich, “Technical milestone-Medical subject headings used to search the biomedical literature,” Journal of the American Med Info. Asso., vol. 8(4), 2001, pp. 317-323, doi:10.1136/jamia.2001.0080317.

[10] I. Whitmore, “Terminologia Anatomica: new terminology for the new anatomist,” Anat Rec., vol. 257(2), 1999, pp. 50-3, doi: 10.1002/(SICI)1097-0185(19990415)257:2<50::AID-AR4>3.0.CO; 2-W.

[11] C. J. O. Baker, A. Shaban-Nejad, X. Su, V. Haarslev, and G. Butler, “Semantic web infrastructure for fungal enzyme biotechnologists,” Journal of Web Semantics, vol. 4(3), 2006, pp. 168-180, doi: 10.1016/j.websem.2006.05.001.

[12] P. W. Crous, “Plant pathology is lost without taxonomy,” Outlooks on Pest Management, vol 16, 2005, pp. 119-123.

[13] K. Al-Aboud, K. Al-Hawsawi, V. Ramesh, D. Al-Aboud, and A. AL-Githami, “An Appraisal of Terms Used in Dermatology. SKINmed, vol. 2(3), 2003, pp. 151-153.

[14] M. T. Cushion, and J. R. Stringer, “Has the Name Really Been Changed? It Has for Most Researchers,” Clinical Infectious Diseases vol. 41, 2005, pp. 1756-1758, doi: 10.1086/498158.

[15] F.C. Odds, T. Arai, A. C. Di Salvo, E. G. V. Evans, R. J. Hay, H. S. Randhawa, M. G. Rinaldi, and T. J. Walsh, “Nomenclature of fungal diseases. A report from a Sub-Committee of the Intl’ Society for Human and Animal Mycology (ISHAM),” J. Med Vet Mycol. Vol. 30(1), 1992, pp. 1-10, doi:10.1080/02681219280000021.

[16] F. C. Odds, and M. G. Rinaldi, “Nomenclature of fungal diseases. Curr. Top. Med. Mycol. Vol. 6, 1995, pp.33-46.

[17] M. Capobianco, C. I. Chesñevar, and G. R. Simari, “Argumentation and the Dynamics of Warranted Beliefs in Changing Environments,” Autonomous Agents and Multi-Agent Systems, vol. 11(2), 2005, pp. 127-151, doi: 10.1007/s10458-005-1354-8.

[18] D. Bryant, and P. Krause, “A review of current defeasible reasoning implementations,” Knowledge Eng. Review, vol. 23(3), 2008, pp. 227-260, doi:10.1017/S0269888908001318

[19] M. R. Capobianco, C. I. Chesñevar, and G. R. Simari, “On the construction of Dialectical Databases,” Inteligencia artificial, vol. 35, 2007, pp. 89-100.

[20] S. Eilenberg, and S. Mac Lane, “General Theory of Natural Equivalences,” T Am Math Soc, vol. 58, 1945, pp. 231-294.

[21] A. Asperti, and G. Longo, “Categories, Types, and Structures: An Introduction to Category Theory for the Working Computer Scientist,” MIT Press, 1991.

[22] R. Heckel, “Graph Transformation in a Nutshell,” Electr. Notes Theor. Comput. Sci. vol. 148(1), 2005, pp.187-198.

[23] F. Drewes, B. Hoffmann, and D. Plump, “Hierarchical Graph Transformation,” J. Comput. Syst. Sci. vol. 64(2), 2002, pp. 249-283, doi:10.1006/jcss.2001.1790.

[24] G. Taentzer, “Distributed Graphs and Graph Transformation,” Applied Categorical Structures, vol. 7(4), pp. 431-462, doi: 10.1023/A:1008683005045.

[25] G. Engels, and A. Schürr, “Encapsulated hierarchical graphs, graph types, and meta types,” Electr. Notes Theor. Comput. Sci. vol. 2, pp. 101-109, doi:10.1016/S1571-0661(05)80186-0.

[26] G. Taentzer, I. Fischer, M. Koch, V. Volle, “Distributed Graph Transformation with Application to Visual Design of Distributed Systems”, In: Ehrig H, Kreowski HJ et al. (eds.) Handbook of Graph Grammars and Computing by Graph Transformation, Vol 3, World Scientific 1999.

[27] H. J. Schneider, “Describing distributed systems by categorical graph grammars,” Proc of 15th int'l workshop on Graph-Theoretic Concepts in Computer Science (WG 89), Springer, LNCS 411, 1989, pp. 121-135, doi:10.1007/3-540-52292-1_9.

[28] A. Shaban-Nejad, and V. Haarslev, “An Abstract Representation Model for Evolutionary Analysis of Multi-Agent Interactions,” Proc. of 2011 IEEE Congress on Evolutionary Computation (IEEE CEC 2011), New Orleans, USA, June 5-8, 2011, pp. 2002-2009, doi: 10.1109/CEC.2011.5949861

[29] G. Taentzer, “Hierarchically Distributed Graph Transformation,” Proc. of 5th Int'l Workshop on Graph Grammars and Applications (TAGT 94), Springer, LNCS 1073, 1994, pp. 304-320, doi: 10.1007/3-540-61228-9_95.

[30] H. Ehrig, F. Orejas, and U. Prange, “Categorical Foundations of Distributed Graph Transformation,” Proc. of 3rd int'l Conference on Graph Transformations (ICGT 06), Springer, LNCS 4178, 2006, pp. 215-229, doi: 10.1007/11841883_16.

[31] H. Ehrig, M. Korff, M. Löwe, “Tutorial Introduction to the Algebraic Approach of Graph Grammars Based on Double and Single Pushouts,” Proc. of 4th Int'l Workshop Graph-Grammars and Their App., Springer, LNCS 532, 1990, pp. 24-37, doi: 10.1007/BFb0017375.

[32] A. Habel, R. Heckel, and G. Taentzer, “Graph Grammars with Negative Application Conditions”, Fundam. Inform. vol. 26(3/4), 1996, pp. 287-313, doi: 10.3233/FI-1996-263404.

[33] G. Busatto, H. J. Kreowski, and S. Kuske, “Abstract hierarchical graph transformation,” Mathematical Structures in Computer Science, vol. 15(4), 2005, pp. 773-819, doi: 10.1017/S0960129505004846

[34] G. Taentzer, “AGG: A graph transformation environment for modeling andvalidation of software,” Proc. of the 2nd International Workshop on Applications of Graph Transformations with Industrial Relevance (AGTIVE 04), Springer, LNCS 3062, 2003, pp. 446-453, doi: 10.1007/978-3-540-25959-6_35.

[35] J. Goguen, “A Categorical Manifesto,” Mathematical Structures in Computer Science, vol 1(1), 1991, pp. 49-67, doi: 10.1017/S0960129500000050.