

# Operations on RDF(S) Ontologies: A Category Theoretic Approach

S. Aliyu<sup>1</sup>, A. F. Donfack Kana<sup>1</sup>, S.B. Junaidu<sup>1</sup>

<sup>1</sup> Department of Mathematics, Faculty of Science, Ahmadu Bello University, Zaria  
{aliyusalisu, dkana, sahalu}@abu.edu.ng

**Abstract.** Resource Description Framework (RDF) provides the foundation for the representation of information on the Semantic Web. Because of the limitation of RDF, RDF(S) was developed to overcome those limitations by adding the RDF schema (RDFS) layer on RDFS with formal support for features such as classes, subclasses and predicate subtyping. In this paper, we showed how RDF(S) Ontologies can (1) be modeled using Category Theory (2) leverage Category Theoretic constructs and operations in issues concerning the formal description and automation of operations on RDF(S) Ontologies e.g. alignment, merging, satisfiability test etc. This has a direct benefit as some major problem areas in ontology engineering such as re-use of ontologies and recombination of ontological modules can be reduced to their mathematical equivalent hence incorporating precision in the process.

## 1 Introduction

Ontologies play an increasingly important role in various areas of Knowledge Representation. Knowledge Representations on the web that makes them accessible and process-able by machines describes the Semantic Web vision. The Semantic Web standard provides a number of Ontological modelling languages that differ in their level of expressivity and are organized in a Semantic Web Stack in such a way that each language level builds on the expressivity of the other. There are several problems when one attempts to use independently developed ontologies. More often, multiple Ontologies are combined which has raised the ontological composition problems. As noted in [2], ontology composition problems is widely seen as both a crucial issue for the realization of the semantic web and as one of the hardest problems to solve. Ontology composition has received wide attention in the research community in recent years [3], [4], [5], [6], [7], [8] and [9]. It was observed in [4] that another important problem to be solved while combining Ontologies is the automation of the process. Techniques that rely strongly on human input are able to score better on precision, but, are less scalable and labor intensive as compared to methods that rely strongly on automation. It was also noted in [2] that the sharing of information based on the intended semantics, is still a fundamental challenge.

It is believed that a formal view on ontologies can contribute a lot in solving ontology composition problems. This paper uses category theory to model the RDF(S) (i.e. the combination of RDF instance data and the RDF Schema (RDFS) layer) Ontologies as a step towards the formalization of possible ontological operations.

The rest of this paper is structured as follows: Section 1.1 introduces RDF and RDFS. Section 1.2 describes some possible operations on ontologies. Section 1.3 describes Category Theory, its associated constructs and operations we employed. Section 2 describes the modeling of our RDF graph Categorically. Section 3 describes the modeling of the RDFS layer Categorically. Section 4 describes the RDF(S) Vocabulary, Interpretation and Semantic Model Categorically. Section 5 discusses the categorical formalization of operations on RDF(S) ontologies. Section 6 discusses the related works and finally, Section 6 gave a brief conclusion to this paper.

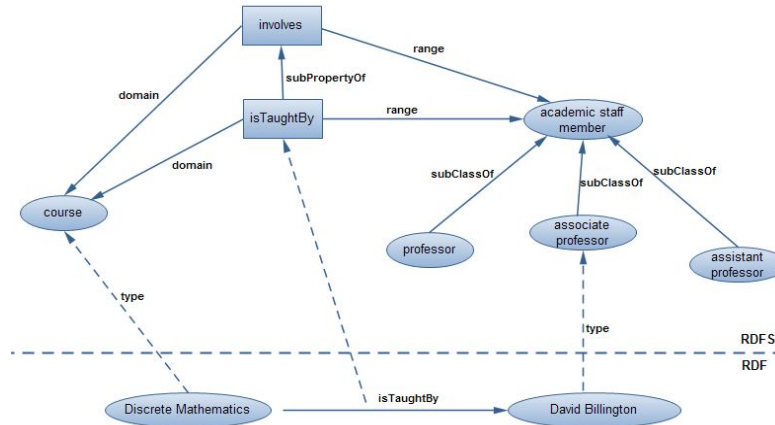
## 1.1 RDF and RDFS

The Resource Description Framework (RDF) is a framework for representing information on the Web [10]. The core structure of the abstract syntax is a set of triples, each consisting of a *subject*, a *predicate* and an *object*. A set of such triples is called an *RDF graph*. An RDF graph can be visualized as a node and directed-arc diagram in which each triple is represented as a node-arc-node link [10]. According to [2], RDF is a universal language that lets users describe resources using their own vocabularies. RDF does not make assumptions about any particular application domain, nor does it define the semantics of any domain. It is up to the user to do so in RDF Schema (RDFS). RDFS is a language for defining the semantics of a particular domain. Example 1 below adopted [2] describes the distinction between RDF and RDFS.

**Example 1:** Consider the RDF statement:

"Discrete Mathematics is taught by David Billington".

The schema for this statement may contain classes such as lectures, academic staff members, first-year courses, and properties such as is taught by, involves, phone, employee ID. Figure 1 illustrates the layers of RDF and RDF Schema. Blocks in Figure 1 represents properties, ellipses above the dashed line represents classes (i.e. Class Names), and ellipses below the dashed line represents instances.



**Fig 1:** RDF and RDFS Layer

The schema in figure1 is itself written in a formal language, RDF Schema, that can express its ingredients such as subClassOf, Class, Property, subPropertyOf, Resource etc.



## 1.2 Some Ontological Operations



It is possible that one application uses multiple RDF(S) Ontologies, especially when we need to integrate with ontologies used by other systems. In this case, some operations may be needed in order to work with all of them. We will summarize some of these operations as described in [1], [3], [5], [14], [16], [17], [18]. The terminologies in this areas is still not stable and different authors may use these terms in a bit shifted meaning, and so the terms may overlap, however, all of these operations are important for the maintenance and integration of knowledge described in RDF(S).

- a) **Instantiation:** Instantiation is the process of determining if an arbitrarily chosen object or concept  $x$  is used in a knowledge model or determining if a given pair of individual  $x$  and  $y$  are instances of a binary relation  $R$  denoted by  $R(x, y)$  i.e. a statement in a knowledge model. Instantiation can be used for query answering.
- b) **Satisfiability:** is an operation for determining when truth is preserved by transformations of RDF(S) ontology.
- c) **Concept Equivalence:** Two concepts  $x$  and  $y$  in a knowledge model are said to be equivalent if  $I(x)=I(y)$  for an Interpretation function  $I$ .
- d) **Concept Disjointness:** Two concepts  $x$  and  $y$  in a knowledge model are said to be disjoint if  $I(x) \neq I(y)$  for the interpretation function  $I$ .
- e) **Subsumption/Entailment:** Let  $C$  and  $D$  be two RDF(S) Ontologies.  $D$  subsumes  $C$  denoted as  $C^I \subseteq D^I$  if  $D$  contains  $C$ . i.e. is an operation for determining whether one knowledge model description is more general than another one.
- f) **Alignment:** the task of establishing a collection of binary relations between concepts that are semantically correspondent to one another.
- g) **Merging:** the task of combining two or more knowledge model producing a new one that embodies the semantic differences and collapses the semantic intersection between the original ones.

## 1.3 Category Theory: Constructs and Operations

**Definition 1 (A Category):** A Category  $C$  is a structure  $(O, M, f, o, id)$ , where:

- $O$  is a collection of objects.
- $M$  is a collection of morphisms defined such that  $g \in M$  iff  $g:A \longrightarrow B$  where  $A,B \in O$ .
- $f: M \longrightarrow O \times O$  is an operation which associates to each morphism its domain object and its codomain object.
- $o$  is an associative operation of morphism composition.
- $id$  is a collection of identity morphism for each object of  $O$ .

In otherwords, a Categorical structure as described by [11] is built on the following constituents.

- A collection of things called *objects*, denoted by  $A, B, C, \dots$  varying over objects.
- A collection of arrows called *morphisms* or *maps*, usually denoted by:  $f, g, h, \dots$  varying over morphisms.
- A relation on morphisms and pairs of objects, called *typing* of the morphisms. For morphism  $f$  and objects  $A, B$ , the relation is denoted  $f: A \longrightarrow B$ . We also say that  $A \longrightarrow B$  is the type of  $f$  and that  $f$  is a morphism from  $A$  to  $B$ .  $A$  and  $B$  are referred to as domain and codomain of  $f$  respectively.

For each pair of morphisms say:  $f: A \longrightarrow B$  and  $g: B \longrightarrow C$  a composite map holds:  $g \circ f: A \longrightarrow C$

~~A Categorical structure must also satisfy the following rules, laws or axioms.~~

A Categorical structure must also satisfy the following rules, laws or axioms.

**Identity Laws:** if  $f: A \longrightarrow B$ , then  $1_B \circ f = f$  and  $f \circ 1_A = f$

**Associative law:** if  $f: A \longrightarrow B$ ,  $g: B \longrightarrow C$  and  $h: C \longrightarrow D$ , then  $(h \circ g) \circ f = h \circ (g \circ f)$

**Remark 1:** It is important to note here that the objects do not have to be sets and the arrows need not be functions [12].

**Definition 2 (subcategory):** A subcategory  $D$  of a category  $C$  is a category for which the following holds.

- All the objects of  $D$  are objects of  $C$  and all the arrows of  $D$  are arrows of  $C$ .
- The source and target of an arrow of  $D$  are the same as its source and target in  $C$ .
- If  $A$  is an object of  $D$  then its identity arrow  $\text{id}_A$  in  $C$  is in  $D$ .
- If  $f: A \longrightarrow B$  and  $g: B \longrightarrow C$  in  $D$ , then the composite  $g \circ f$  (in  $C$ ) is in  $D$  and is the composite in  $D$ . [13]

In otherwords, being a subcategory requires the objects and arrows of the subcategory to be identical with some of the objects and arrows of the category containing it.

**Definition 3 (Hom Set) :** If  $A$  and  $B$  are two objects of a category  $C$ , then the set of all arrows of  $C$  that have source  $A$  and target  $B$  is denoted as  $\text{Hom}_C(A, B)$ . A set of the form  $\text{Hom}_C(A, B)$  is called the **hom set**. [13]

In otherwords, a set that contains all the morphisms between one object and another is the hom set of the two objects where the first object is the domain and the second is the codomain.

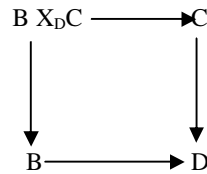
Operations in category theory such as pullback, coproduct and pushout which may be vital in our formalization process are defined below as described in [15].

**Definition 4 (Pullback):** Let  $B$ ,  $C$ , and  $D$  be sets, and let  $f: B \longrightarrow D$  and  $g: C \longrightarrow D$  be functions. The pullback of  $B \longrightarrow D \longleftarrow C$ , denoted  $B \times_D C$ , is defined to be the set of pairs  $(b, c)$  such that  $f(b)=g(c)$  i.e.

$$B \times_D C := \{ (b, c) \mid b \in B, c \in C, \text{ and } f(b)=g(c) \}$$

together with the obvious maps  $B \times_D C \longrightarrow B$  and  $B \times_D C \longrightarrow C$ , which maps an element  $(b, c)$  to  $b$  and to  $c$ , respectively.

i.e. the pullback of  $B \xrightarrow{f} D \xleftarrow{g} C$  is a commutative square.



So, we write  $A = B \times_D C$  is interpreted as "A is the pullback of B and C over D."

**Definition 5 (Coproducts):** Given sets  $A$  and  $B$ , their *coproduct*, denoted  $A + B$ , is the set

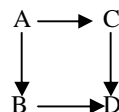
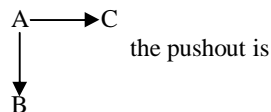
$$A + B = \{ (a') \mid a \in A \} \cup \{ (b') \mid b \in B \}.$$

In other words, coproducts are also called "disjoint unions." If  $A$  and  $B$  are sets with no members in common, then the coproduct of  $A$  and  $B$  is their union. However, if they have elements in common, one must include both copies in  $A+B$  and differentiate between them.

**Definition 6 (Pushout Operation):** Let  $A$ ,  $B$ , and  $C$  be sets and let  $f: A \longrightarrow B$  and  $g: A \longrightarrow C$  be functions. The *pushout* of  $B \longleftarrow A \longrightarrow C$ , denoted  $B +_A C$ , is the quotient of  $B + C$  by the equivalence relation generated by declaring  $b \sim c$  (i.e.  $b$  is equivalent to  $c$ ) if:  $b \in B$ ,  $c \in C$ , and there exists  $a \in A$  with  $f(a) = b$  and  $g(a) = c$ . [12]

Pushouts can express unions in which an overlap is declared. They can also express "quotients," where different objects can be declared equivalent. Given three objects and two arrows arranged as to the left, the pushout is drawn as the commutative square to the right:

Given,



We write  $D = B +_A C$  interpreted as “ $D$  is the pushout of  $B$  and  $C$  along  $A$ .” The idea is that an instance of the pushout  $B +_A C$  is any instance of  $B$  or any instance of  $C$ , but where some instances are considered equivalent to others.

**Definition 7 (Subobject):** A Subobject of an object  $X$  in a Category  $C$  is a mono  $m: M \longrightarrow X$  such that the image subset  $\{m(y) \mid y \in M\} \subseteq X$ . [12]

We can think of the mono  $m: M \longrightarrow X$  itself as determining a "part" of  $X$

**Definition 8 (Slice Category):** If  $C$  is a category and  $A$  any object of  $C$ , the slice category  $C/A$  is described this way:

- An object of  $C/A$  is an arrow  $f: D \longrightarrow A$  of  $C$  for some object  $D$ .
- An arrow of  $C/A$  from  $f: D \longrightarrow A$  to  $g: D' \longrightarrow A$  is an arrow  $h: D \longrightarrow D'$  with the property that  $f = g \circ h$ . [13]

**Definition 9 (Submorphism):** Given subobjects  $m$  and  $m'$  of an object  $X$  in a Category  $C$ , we say  $m \subseteq m'$  iff there exist a morphism  $f: m \longrightarrow m'$  in  $C/X$  [12].

## 2 Categorical Model of RDF Graph

**Definition 10 (A Categorical RDF ( $C_{rdf}$ )):** A Categorical RDF Ontology  $C$  is a structure  $(R, P, f, o, id)$ , where:

- $R$  is a collection of resources;
- $P$  is a collection of morphisms.
- $f: P \longrightarrow W$ , where  $W \subseteq R \times R$ ,  $f$  is an operation which associates to each morphism its domain object and codomain object;
- $\circ$  is an associative operation of morphism composition;
- $id$  is a collection of identity morphisms for each object of  $R$ .

**Proposition 1.0** an RDF graph is a Categorical Structure.

*Proof:*

### For Identity

By our definition, we have collection of objects which are resources and collection of morphisms which are properties.

Consider a resource which is an object  $A \in R$ , since  $A$  has a self element, then the morphism

$f: A \longrightarrow A$  is unique which is the identity for  $A$ . Thus Identity exist.

### For Associativity

Let  $f: A \rightarrow B$ ,  $g: B \rightarrow C$  and  $h: C \rightarrow D$  be **RDF comments**. Composing  $f$  and  $g$  we have  $f \circ g: A \rightarrow C$  and  $f \circ g$  with  $h$  we will get  $(f \circ g) \circ h: A \rightarrow D$  and  $\forall x \in A: ((f \circ g) \circ h)(x) = (f \circ g) h(x) = f(g(h(x)))$  ----- 1.0

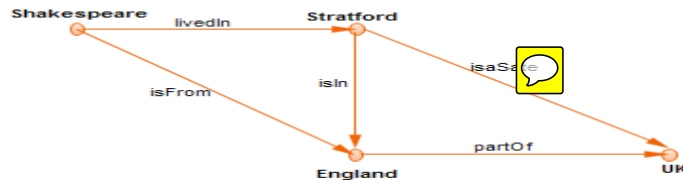
Now consider, composing  $g$  with  $h$  we have  $g \circ h: B \rightarrow D$  and composing  $f$  with  $(g \circ h)$  we have  $f \circ (g \circ h): A \rightarrow D$  and  $\forall x \in A$ ,

$$(f \circ (g \circ h))(x) = f(g \circ h)(x) = f(g(h(x))) \text{ ----- 1.1}$$

Since, 1.0 equals 1.1, thus associativity is ascertained.

### Example 2

Consider the RDF graph:



The statement "Shakespeare *livedIn* Stratford" can be expressed categorically as:  
 $livedIn(Shakespeare) = Stratford$  or  $livedIn: Shakespeare \rightarrow Stratford$

For the sake of simplicity, let the following notations represent the instance data as depicted by the graph:

$c_1 = Shakespeare, c_2 = Stratford, c_3 = England, c_4 = UK$

$p_1 = livedIn, p_2 = isIn, p_3 = partOf, p_4 = isFrom, p_5 = isaState$

Hence,  $R = \{c_1, c_2, c_3, c_4\}$  and  $P = \{p_1, p_2, p_3, p_4, p_5\}$  from Definition 5.

For the associative property of the composition operator, It will be observed that

$$(p_3 \circ (p_2 \circ p_1))(c_1) = c_4$$

Which is equivalent to saying:

$$(partOf \circ (isIn \circ livedIn))(Shakespeare) = UK$$

On the other hand, also consider:

$$((p_3 \circ p_2) \circ p_1)(c_1) = c_4$$

Which is equivalent to saying:

$$((partOf \circ isIn) \circ livedIn)(Shakespeare) = UK$$

hence the associative property is ascertained.

## 3 Categorical Model of RDFS

We now define the RDFS layer categorically in definition 6.

**Definition 6 (A Categorical RDFS ( $C_{rdfs}$ )):** A Categorical RDFS is a structure  $(R_C, P_C, F, o, id)$ , where:

- $R_C$  is a collection of classes of resources (rdfs: Resources)

- $P_C$  is a collection of both classes and instances of morphisms (rdf: Property)
- $F$  is a set of two operations  $f$  and  $g$  such that  $f$  is an operation that assigns to an element of  $P_C$  its domain  $dom$  and codomain which are  $f: P_C \longrightarrow W$ , where  $W \subseteq R_C \times R_C$  and  $g$  is an operation which associates to each morphism its domain object (an element of  $P_C$ ) and codomain object (an element of  $P_C$ ) e.g. rdf: subPropertyType
- $o$  is a composition operation of morphism which is associative.
- $id$  is a collection of identity morphisms for each object of  $R$ .

### Remarks 2:

**Classes:** Elements of  $R_C$  of  $C_{rdfs}$  denotes class names for the schema. Mappings from  $R_C$  to  $R_C$  denotes the relationship between classes. In other words, the objects of the Category  $C_{rdfs}$  represents our schema class names.

**SubClasses:** Let  $C_1$  and  $C_2$  represent two objects of  $C_{rdfs}$ ,  $C_1$  is a subclass of  $C_2$  iff  $C_1$  is a subobject of  $C_2$ .

**Properties:** All morphisms that exist in  $C_{rdfs}$  denotes properties that exist between objects (class representation).

**SubProperties:** Let  $h$  and  $h^1$  be two morphisms (properties) in  $C_{rdfs}$ ,  $h$  is a sub property of  $h^1$  iff  $h$  is a submorphism of  $h^1$ .

RDF Schema extends RDF to include a larger vocabulary. RDFS provides a new semantic construct, a *class*, which is a resource that represents a set of things in the universe. So, two classes can be related to one another (e.g. *rdfs: subclassOf*) or two properties relating to each other (e.g. *rdfs: subPropertyOf*). Consider the information in Table1. *course* and *associate professor* defines the schema (RDFS) information (i.e. classes) while values beneath each class represents an RDF instance data. So, with an RDFS, we can make multiple statements instances that fits the schema information.

By RDF(S) we mean the combination of both RDF instance data definition and the RDFS schema definition layer.

**Table 1:** Tabular Illustration of RDF(S) Ontological Model with RDF Instance Data and RDFS Schema Information.

<i>isTaughtBy</i>	
<b>course</b>	<b>associate professor</b>
Discrete Mathematics	David Bilington
Simulation	Jerry Banks
Database Systems	Elmasri
Category Theory	Steve Awodey



**Proposition 2.0** an RDFS Model is a Categorical Structure.

*Proof:*

The proof here follows the same pattern as that for proposition 1.0.

### Example 3

Consider the RDFS Ontology in figure 1. From Definition 6:

$R_C = \{\text{course, professor, associate professor, academic staff member, assistant professor}\}$

$P_C = \{\text{involves, isTaughtBy, domain, subClassOf, subPropertyOf}\}$

Performing the  $f$  operation on this set of vocabulary, the following statements can be generated:

professor *subClassOf* academic staff member  
professor *subClassOf* professor  
assistant professor *subClassOf* academic staff member  
associate professor *subClassOf* academic staff member

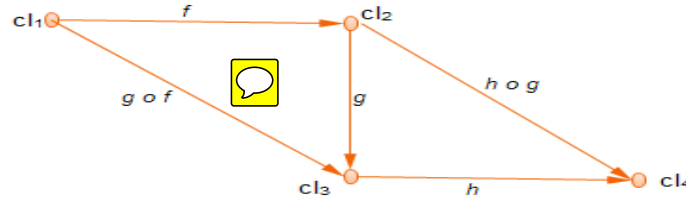
The statement: professor *subClassOf* professor, serve as a proof for the identity property of the RDFS Categorical structure. Also note that not all statements generated may be meaningful.

Performing the  $g$  operation also on this set of vocabulary can generate the statements:

isTaughtBy *subPropertyOf* involves  
isTaughtBy *subPropertyOf* isTaughtBy etc.

The statement: isTaughtBy *subPropertyOf* isTaughtBy, serve as a proof for the identity property of the RDFS Categorical structure.

For the associative property of the composition operator, let  $cl_1, cl_2, cl_3, cl_4$  represent classes (objects) in an RDFS Categorical structure. If the mappings between the class names is as shown in the categorical structure below:



**Fig. 2.** RDF Schema (RDFS) representing mappings between class representations

From the diagram we can compare  $h \circ (g \circ f)$  and  $(h \circ g) \circ f$ . It will turn out that these two functions are always identical i.e.

$$(h \circ (g \circ f)) = ((h \circ g) \circ f)$$

Since for any  $a \in cl_1$ , we have

$$(h \circ (g \circ f))(a) = h(g(f(a))) = ((h \circ g) \circ f)(a)$$

So, the two functions are equal since for every argument they have the same value. Hence the associative property.

From definition 5 and 6, the whole bunch of vocabulary to be used are gotten from the union of the sets  $R$ ,  $P$ ,  $P_C$  and  $R_C$ . In order not to reinvent the wheel, vocabulary used in this model will be those provided by RDF(S) Syntax specification. The semantics to these syntax will be based on the RDF(S) Semantics specification expressed by our Interpretation model categorically.

#### 4 Categorical RDF(S) Vocabulary Model

**Definition 7 (A Categorical RDF(S) Vocabulary ( $C_V$ )):** A Categorical RDF(S) Ontology is a 3-tuple structures  $\langle C_{rdfs}, C_{rdf}, F \rangle$  where:

- $C_{rdfs}$  is a Categorical RDFS Model.
- $C_{rdf}$  is a Categorical RDF Model.
- $F$  is a functor ( $rdf: type$ ) that maps a  $C_{rdf}$  object or property to a  $C_{rdfs}$  objects or property.

In otherwords,  $C_{rdf}$  represents the RDF layer while  $C_{rdfs}$  describes the RDFS layer in Figure 1. The functor defines the rdf typing property ( $rdf: type$ ) that associates to each RDF object its schema object.

**Definition 8 (A Categorical RDF(S) Ontology Interpretation Model ( $C_I$ )):** A Categorical RDF(S) Ontology Interpretation Model  $C_I$  is a structure  $\langle R_C^I, R^I, P^I, F_I \rangle$  where  $R_C^I$  is a collection of all referred classes,  $R^I$  is a collection of Resources representing referred resources,  $P^I$  is a collection representing referred properties and  $F_I$  is a collection of four operations ( $f_1, f_2, f_3, f_4$ ) as described below:

- $\langle Z, P^I, f_1 \rangle$  where
  - $Z$  is the powerset of  $R^I \times R^I$  (i.e. Cartesian product of referred resources).
  - $P^I$  is the collection of referred properties.
  - $f_1$  is an operation that assigns an element of  $P^I$  to an ordered pair from  $Z$  denoted by  $f_1: P^I \rightarrow Z$ .
- $\langle Y, R^I, f_2 \rangle$  where
  - $Y$  is the union of referred resources and properties,  $R^I \cup P^I$ .
  - $R^I$  is a collection of referred resources.
  - $f_2$  is an operation that maps a resource into the union of referred resources and properties denoted by  $f_2: R^I \rightarrow Y$ .
- $\langle L, R^I, f_3 \rangle$  where
  - $L$  represents literals where  $L \subset IRIs$
  - $R^I$  is the collection of resources.


- $f_3$  is an operation that maps partially a literal into a collection of resources denoted by  $f_3 : L \longrightarrow \mathbf{R}^I$ .
- $\langle R_C^I, R^I, f_4 \rangle$  where
  - $R_C^I$  represents set of referred classes.
  - $R^I$  is the collection of resources.
  - $f_4$  is an operation that maps from  $R_C^I$  to the set of subsets of  $R^I$  by  $f_4 : R_C^I \longrightarrow \mathbf{R}^I$ .

**Remark 3:** Definition 8 was achieved by modeling the simple interpretation provided in [14] categorically.

Now, having defined our interpretation model, the function from any Categorical RDF(S) expression or statement (i.e. vocabulary) into our Interpretation model describes the semantic model.

**Definition 9 (A Categorical RDF(S) Ontology Semantic Model):** A Categorical Ontology Semantic Model is a 3-tuple structures  $\langle C_V, C_I, \mathcal{I} \rangle$  where:

- $C_V$  is a Categorical RDF(S) Ontology
- $C_I$  is the Interpretation Model.
- $\mathcal{I}$  is the interpretation function (i.e. functor).

An Interpretation provides just enough information about a possible way the world might be in order to fix the truth-value (true or false) [14]. So, in order to determine the truthfulness of an RDF(S) graph, we define as rules in **Algorithm 1** the interpretation process which also describes the satisfiability test. 

Let  $C$  be a Categorical RDF(S) Ontological graph consisting of classes, objects, triples or statements, the interpretation of the graph is defined as follows. Note that this rules changes as the expressivity of our ontology language changes.

---

**ALGORITHM 1** *SatisfiabilityTest*( $E, C, C_I$ )

---

```
// Implements a Satisfiability test for an expression (literal, statement or a graph)
//Input: an expression E and a Categorical RDF(S) Ontology C, Interpretation
Model CI
//Output: Boolean true or false
if E is a literal
  I(E)  $\longleftarrow$  LiteralInterpreter(E, CI)
  if I(E)  $\neq \emptyset$ 
    return true
  else
    return false
else if E is a resource
  I(E)  $\longleftarrow$  ResourceInterpreter(E, CI)
  if I(E)  $\neq \emptyset$ 
    return true
  else
```

```

        return false
    else if E is a Class
        I(E) ← ClassInterpreter(E, CI)
        if I(E) ≠ ∅ return true
        else return false
    else if E is a statement with <s, p, o>
        I(p) ← PropertyInterpreter(p, CI)
        if (I(p) ≠ ∅) && (<I(s), I(o)> ∈ StatementInterpreter(I(p), CI))
            return true
        else return false
    else // i.e. when we have multiple triples (say in graph, G)
        foreach El in G // where El ∈ G
            if I(El)=false then
                I(E)=false
            else I(E)=true
        LiteralInterpreter(E, CI)
        foreach r in Rl
            if I(E)=r
                return r
            else
                return undefined
        ResourceInterpreter(E, CI)
        Rp ← Rl ∪ Pl
        foreach q in Rp
            if I(E) = q
                return q
            else
                return undefined
        PropertyInterpreter(p, CI)
        foreach t in Pl
            if I(p) = t
                return t
            else
                return undefined
        StatementInterpreter(I(p), CI)
        foreach I(p) in CI
            return <I(s), I(o)>
        else
            return undefined
    ClassInterpreter(c, CI)
    foreach s in CRl
        if I(c) = s
            return s
        else return undefined

```

---

## 5 Categorical Formalization of Operations on RDF(S) Ontologies

**Definition 10 (Concept Instantiation):** Let  $F(D,E)$  be a Categorical RDF(S) ontology,  $Ob(D)$  be the set of objects in  $D$  (Categorical rdf graph) and  $Ob(E)$  be the set of objects in  $E$  (Categorical rdfs graph), if  $c_1 \in Ob(D)$  and  $c_2 \in Ob(E)$  then  $c_2$  is a concept instance of  $c_1$ , if  $F(c_1)=c_2$ . (Note:  $F$  represents the *rdf:type*)

For example, consider the RDF(S) ontology in Figure1.  
Let  $D$  be the Categorical RDF model and  $E$  be the Categorical RDFS model.

Clearly,  
 $Ob(D)=\{\text{Discrete Mathematics, David Billington}\}$   
 $Ob(E) = \{\text{course, academic staff member, professor, associate professor, assistant professor}\}$

For simplicity let's use the notations:

$Ob(D)=\{d_1, d_3\}$  and  $Ob(E)=\{e_1, e_2, e_3, e_4, e_5\}$ , since  $F(d_1)=e_1$  then  $d_1$  is a concept instance of  $e_1$ .

**Definition 11 (Relation Instantiation):** Let  $F(D,E)$  be a Categorical RDF(S) ontology,  $Ob(D)$  be the set of objects in  $D$  (Categorical rdf graph) and  $Ob(E)$  be the set of objects in  $E$  (Categorical rdfs graph), if  $d_1, d_2 \in Ob(D)$  and  $e_1, e_2 \in Ob(E)$  then  $p:d_1 \longrightarrow d_2$  is a relation instance of  $F(D,E)$  iff  $Fp \in \text{Hom}(Fd_1, Fd_2)$  where  $Fd_1=e_1$  and  $Fd_2=e_2$ .

For example, consider the statement:

*Discrete Mathematics isTaughtBy David Billington.*

Let,

$s = \text{Discrete Mathematics}$ ,  $p = \text{isTaughtBy}$  and  $o = \text{David Billington}$

Clearly, the  $\text{Hom}(F(s), F(o)) = \{\text{isTaughtBy}\}$  and  $F(p) = \text{isTaughtBy} \in \text{Hom}(F(s), F(o))$ , hence the statement is a relation instance of the Ontology.

An algorithm to test for concept or relation instantiation is defined in Algorithm 1.

---

### ALGORITHM 1 *InstantiationChecker*( $\alpha$ , $C$ )

---

```
// Algorithm for checking both concept and relation instantiation
//Input: an expression  $\alpha$  (which could be a concept or a relation i.e. statement), a
//Categorical RDF(S) Ontology  $C:=F(D,E)$ 
//Output: Boolean true or false
 $Ob(D) = \emptyset$  // Initializing our collection of objects
 $Ob(E) = \emptyset$  // Initializing our collection of objects
for any object  $o^*$  in  $D$ 
     $Ob(D) \leftarrow Ob(D) \cup \{o^*\}$ 
```

```

for any object  $o^{**}$  in  $E$ 
   $Ob(E) \leftarrow Ob(E) \cup \{o^{**}\}$ 
//checking concept instance
if  $\alpha$  is a concept or resource
  foreach  $o^*$  in  $Ob(D)$ 
    if  $F(o^*) \in Ob(E)$ 
      return true
    else
      return false
//checking relation instance
else if  $\alpha$  is a statement with  $\langle s, p, o \rangle$ 
  foreach edge  $e^*$  in  $Hom_D(s, o)$  do
    if  $F(e^*) \in Hom_E(F(s), F(o))$ 
      return true
    else
      return false

```

---

**Definition 12 (Satisfiability):** Let  $C$  be a Categorical RDF(S) Ontology,  $C_1$  be the Categorical Interpretation Model of  $C$  and  $I$  be a functor from  $C$  to  $C_1$ . An expression,  $E$  is satisfiable if  $I(E): C \longrightarrow C_1$  exist. Where  $E$  could be a literal, resource or a statement  $\langle s, p, o \rangle$ .

A satisfiability test will return a true value if the interpretation exist or a false if it does not exist. Algorithm 1 defines a satisfiability test for either a concept or a relation.

**Definition 12 (Concept Equivalence) :** Let  $C$  be a Categorical RDF(S) Ontology,  $C_1$  be the Categorical Interpretation Model of  $C$  and  $I$  be a functor from  $C$  to  $C_1$ . Two concepts  $E_1$  and  $E_2$  are equivalent if  $I(E_1) = I(E_2)$ .

Algorithm 3 defines an equivalence test for two concepts.

---

**ALGORITHM 3** *CheckEquivalence*( $E_1, E_2, C, C_1$ )

---

```

// Algorithm for checking concept equivalence
//Input: two expression  $E_1$  and  $E_2$ , a Categorical RDF Ontology  $C$  and its
//Interpretation model  $C_1$ 
//Output: Boolean true or false
if  $E_1$  and  $E_2$  are both literals
   $I(E_1) \leftarrow LiteralInterpreter(E_1, C_1)$ 
   $I(E_2) \leftarrow LiteralInterpreter(E_2, C_1)$ 
  if ( $I(E_1) = I(E_2)$ )
    return true
  else
    return false
if  $E_1$  and  $E_2$  are both resources
   $I(E_1) \leftarrow ResourceInterpreter(E_1, C_1)$ 

```

---

```

I(E2) ← ResourceInterpreter(E2, C1)
if (I(E1) = I(E2))
    return true
else
    return false

```

---

**Definition 13 (Concept Disjointness) :** Let  $C$  be a Categorical RDF(S) Ontology,  $C_1$  be the Categorical Interpretation Model of  $C$  and  $I$  be a functor from  $C$  to  $C_1$ . Two concepts  $E_1$  and  $E_2$  are disjoint if  $I(E_1) \neq I(E_2)$ .  
Algorithm 4 defines a disjointness test for two concepts.

---

**ALGORITHM 4** *CheckDisjointness*( $E_1, E_2, C, C_1$ )

---

```

// Algorithm for checking concept disjointness
//Input: two concept  $E_1$  and  $E_2$ , a Categorical RDF Graph  $C$  and its Interpretation
//model  $C_1$ 
//Output: Boolean true or false
if  $E_1$  and  $E_2$  are both literals
    I(E1) ← LiteralInterpreter(E1, C1)
    I(E2) ← LiteralInterpreter(E2, C1)
    if (I(E1) ≠ I(E2))
        return true
    else
        return false
if  $E_1$  and  $E_2$  are both resources
    I(E1) ← ResourceInterpreter(E1, C1)
    I(E2) ← ResourceInterpreter(E2, C1)
    if (I(E1) ≠ I(E2))
        return true
    else return false

```

---

**Definition 14 (Subsumption) :** Let  $D = F(L, M)$  and  $C = G(N, P)$  be two categorical RDF(S) Ontologies,  $D$  subsumes  $C$  i.e.  $C \subset D$  iff  $N$  is a subcategory of  $L$  and  $P$  is a subcategory of  $M$ .

Algorithm 5 defines a subsumption test for two ontologies.

---

**ALGORITHM 5** *CheckSubsumption* ( $D, C$ )

---

```

// Algorithm for checking ontology subsumption
//Input: two ontologies  $D = F(L, M)$  and  $C = G(N, P)$ ,
//Output: Boolean true or false
if  $L$  is a subcategory of  $N$  &&  $M$  is a subcategory of  $P$ 
    return true
else return false

```

---

**Definition 15 (Alignment):** Let  $P=F(A,B)$  and  $R=G(C,D)$  be two categorical RDF(S) Ontologies and  $Q$  be the Categorical RDF(S) Interpretation model, the alignment operation between  $P$  and  $R$  is the pullback operation

$$P \xrightarrow{I_P} Q \xleftarrow{I_R} R, \text{ denoted as } P \times_Q R \text{ which is defined as:}$$

$$P \times_Q R = \{ ((a, b), (c, d)) \mid I_P(a) = I_Q(c), I_P(b) = I_Q(d) \text{ where } a \in A, b \in B, c \in C, d \in D \}$$

In otherwords, the first pair is equivalent in interpretation to the second pair where the first element in each pair represents the instance data and the second represents the schema or classname.

Algorithm 6 defines the alignment operation between two categorical RDF(S) Ontologies.

---

**ALGORITHM 6** *Align* ( $P, R, Q$ )

---

// Algorithm for alignment operation  
//Input: two ontologies  $P$  and  $R$ , and an Interpretation Model  $Q$ ,  
//Output: set of pairs of semantically corresponding concepts from  $P$  and  $R$   
Result =  $\emptyset$  //set containing pairs of concepts that are semantically equal  
 $X_1 = \emptyset$  //set of objects or concepts of category A in Ontology  $P$   
 $X_2 = \emptyset$  //set of objects or concepts of category B in Ontology  $P$   
 $Y_1 = \emptyset$  // set of objects or concepts of category C in Ontology  $R$   
 $Y_2 = \emptyset$  // set of objects or concepts of category D in Ontology  $R$   
 $Pr(P)$  // set of instance-schema pair in Ontology  $P$   
 $Pr(R)$  // set of instance-schema pair in Ontology  $R$

```

for any object  $o_1^*$  in A
     $X_1 \leftarrow X_1 \cup \{o_1^*\}$ 
for any object  $o_2^*$  in B
     $X_2 \leftarrow X_2 \cup \{o_2^*\}$ 

for any object  $o_1^{**}$  in C
     $Y_1 \leftarrow Y_1 \cup \{o_1^{**}\}$ 
for any object  $o_2^{**}$  in D
     $Y_2 \leftarrow Y_2 \cup \{o_2^{**}\}$ 

foreach  $o_1^*$  in  $X_1$ 
    foreach  $o_2^*$  in  $X_2$ 
        if  $o_1^*$  isInstanceOf  $o_2^*$ 
             $Pr(P) \leftarrow Pr(P) \cup \{(o_1^*, o_2^*)\}$ 

```



```

foreach  $o_1^{**}$  in  $Y_1$ 
  foreach  $o_2^{**}$  in  $Y_2$ 
    if  $o_1^{**}$  isInstanceOf  $o_2^{**}$ 
       $\mathbf{Pr(R)} \leftarrow \mathbf{Pr(R)} \cup \{(o_1^{**}, o_2^{**})\}$ 

foreach  $(o_1^*, o_2^*)$  in  $\mathbf{Pr(P)}$ 
  foreach  $(o_1^{**}, o_2^{**})$  in  $\mathbf{Pr(R)}$ 
    if  $\text{Interpretation}(o_1^*) = \text{Interpretation}(o_1^{**}) \ \&\& \ \text{Interpretation}(o_2^*) = \text{Interpretation}(o_2^{**})$ 
      return  $\mathbf{Result} \leftarrow \mathbf{Result} \cup \{(o_1^*, o_2^*), (o_1^{**}, o_2^{**})\}$ 

return  $\mathbf{Result}$ 

```

---

**Definition 16 (Merging):** Let  $B=G(J,K)$ ,  $C=H(L,M)$  be two RDF(S) categorical ontology and  $A=F(D,E)$  be a RDF(S) categorical Interpretation model, the merge operation between B and C is the pushout operation

$$B \xrightarrow{I_P} A \xleftarrow{I_R} C, \text{ denoted as } B +_A C \text{ which is defined as the}$$

quotient of  $B+C$  by the equivalence relation generated by declaring  $(b_1, b_2) \sim (c_1, c_2)$  if  $(b_1, b_2) \in B$ ,  $(c_1, c_2) \in C$  and there exist  $(a_1, a_2) \in A$  with  $I_P(a_1, a_2) = (b_1, b_2)$  and  $I_R(a_1, a_2) = (c_1, c_2)$  where  $I(a_1, a_2) = (I(a_1), I(a_2))$ .

**Example (Merge Operation via pushout):** Given that  $A=\{(1,i), (2,j), (3,k), (4,l)\}$ ,  $B=\{(m,a), (n,b), (o,c), (u,d), (v,e)\}$  and  $C=\{(w,p), (x,q), (y,r), (z,s), (h,t)\}$  where A is related to C by the function **F** and to B by the function **G** as shown below:

B						
A \		(m, a)	(n, b)	(o, c)	(u, d)	(v, e)
(1, i)			X			
(2, j)				X		
(3, k)					X	
(4, l)						X

**Fig. 3 :** Relationship between A and B

A \ C	C				
	(w, p)	(x, q)	(y, r)	(z, s)	(h, t)
(1, i)	X				
(2, j)		X			
(3, k)			X		
(4, l)				X	

**Fig. 3** : Relationship between A and C

a) The coproduct of B and C is given as:

$$B+C=\{(m,a), (n,b), (o,c), (u,d), (v,e), (w,p), (x,q), (y,r), (z,s), (h,t)\}$$

b) Generating the equivalence relation by declaring  $(b_1,b_2) \sim (c_1,c_2)$  if  $(b_1,b_2) \in B$  ,  $(c_1,c_2) \in C$  and there exist  $(a_1,a_2) \in A$  with  $I_p(a_1,a_2)=(b_1,b_2)$  and  $I_R(a_1,a_2)=(c_1,c_2)$ , then we will get:

$(a_1, a_2) \in A$	$F(b_1, b_2) \sim G(c_1, c_2)$	
(1, i)	(n, b)	$\sim$ (w, p)
(2, j)	(o, c)	$\sim$ (x, q)
(3, k)	(u, d)	$\sim$ (y, r)
(4, l)	(v, e)	$\sim$ (z, s)

**Fig 4:** Generating Equivalence Relation

c) Finally, the quotient of  $B+C$  by the equivalence relation will give the pushout as:

$$B+_AC= (B+C)/\sim \Rightarrow \{ \{(m,a), \{(n,b), (w,p)\}, \{(o,c),(x,q)\}, \{(u,d),(y,r)\}, \{(v,e),(z,s)\}, \{(h,t)\} \}$$

Note that elements or pairs that are equivalent with respect to A are grouped together or merged. So, if B and C describes our categorical RDF(S) ontologies while A describes our interpretation model, it is clear that the merge operation between B and C can be achieved via the pushout operation.

Algorithm 7 defines the merge operation between two categorical RDF(S) Ontologies.

---

**ALGORITHM 7** *Merge ( B, C, A )*

---

```

// Algorithm for merging operation
//Input: two Categorical RDF Ontological Graphs B and C and an Interpretation
//Model A,
//Output: a single ontology D
//Assumption: morphisms will map naturally

D = ∅ //merged ontology with pairs of instance - schema pair
X1 = ∅ //set of objects or concepts of category J in Ontology B
X2 = ∅ //set of objects or concepts of category K in Ontology B
Y1 = ∅ // set of objects or concepts of category L in Ontology C
Y2 = ∅ // set of objects or concepts of category M in Ontology C
Pr(B) // set of instance-schema pair in Ontology B
Pr(C) // set of instance-schema pair in Ontology C
for any object ji in J
    X1 ← X1 ∪ { ji }
for any object ki in K
    X2 ← X2 ∪ { ki }

for any object li in L
    Y1 ← Y1 ∪ { li }
for any object mi in M
    Y2 ← Y2 ∪ { mi }
foreach ji in X1
    foreach ki in X2
        if ji isInstanceOf ki // creating elements of B
            Pr(B) ← Pr(B) ∪ { (ji, ki) }

foreach li in Y1
    foreach mi in Y2
        if li isInstanceOf mi // creating elements of C
            Pr(C) ← Pr(C) ∪ { (li, mi) }
foreach (ji, ki) in Pr(B)
    foreach (li, mi) in Pr(C)
        if InterpretationOf(ji, ki) = InterpretationOf(li, mi)
            D ← D ∪ { (ji, ki) } or { (li, mi) }
D ← D ∪ (Pr(B)-D) ∪ (Pr(C)-D)
return D

```

---

## 6 Related Work

Several other approaches and techniques exist for the effective and efficient management of Ontologies. According to [4], most of these techniques are semi-automatic and are focused at constructing software tools that can be used to combine independently developed ontologies. In contrast, the approach taken in this paper is to develop a systematic mathematical theory for ontological operations management. Other works that have also considered a mathematical approach include the work of [4] which described an algebra for composing ontologies by defining a recursive finite typing of RDF collections for expressing ontologies expressed in heterogeneous languages, then constructing a well-founded algebraic scheme over a universe of ontologies that respects well known algebraic operations and identities. The semantics of his model was defined as well founded set in ZF set theory. Kaushik's work has the limitation of considering only the RDF level in the semantic web stack. [7] defined an algebra of relations in order to express the relationship between ontology entities in a general way. [6] used category theoretical model to express alignments between ontologies. In their approach, they view alignments independently from the language expressing ontologies and from the techniques used for finding the alignments. They introduced a categorical structure, called V-alignment made of a pair of morphisms with a common domain having the ontologies as co-domain. Their approach shows lack of expressivity with respect to complex semantic relationship such as concept and predicate subsumption. Also, [6] viewed an ontology as a categorical object where operations between more than one ontologies are performed structurally with no consideration of the semantics of the data involved. [15] describes a category theoretic model for knowledge representation without any consideration for a particular application domain. Some of his ideas like considering concepts as objects of a categorical structure was employed in our modeling.

## 6 Conclusion

This paper has modeled RDF(S) Ontology using Category Theory. We have captured most of the RDF(S) modeling constructs categorically as we intend to build upwards along the semantic web stack to fully capture the requirements of the Semantic Web and efficiently managing its information represented as ontologies both syntactically and semantically. Ontological statement and operations were then reduced to their mathematical equivalent hence incorporating mathematical precision in the process. Algorithms for possible ways of implementation of some ontological operations were also described.

## References

1. Obitko, M.: *Translation between Ontologies in Multi-Agent Systems*. Prague: Czech Technical University, Faculty of Electrical Engineering. (2007)
2. Antoniou, Gregor: *A Semantic Web Primer*. United States of America: Massachusetts Institute of Technology (2008)
3. Isabel Cafezeiro and Edward Hermann Haeusler: Semantic Interoperability via Category Theory (2007) Proceeding: ER '07 Tutorials, posters, panels and industrial contributions at the 26th international conference on Conceptual modeling (2007) Volume 83,197-202
4. Kaushik, S., Farkas, C., Wijesekera, D. and Ammann, P.: An Algebra for Composing Ontologies (2008)
5. Klein, Michel: Combining and relating ontologies: an analysis of problems and solutions. *Proceedings of the 17th International Joint Conference on Artificial Intelligence* (pp. 53-62). Seattle: Workshop: Ontologies and Information Sharing (2001)
6. Zimmermann, A., Krötzsch, M., Euzenat, J. and Hitzler, P.: Formalizing Ontology Alignment and its Operations with Category Theory. *Proceedings of the 4th International Conference on formal Ontology in Information Systems*. Baltimore, Maryland, USA: IOS Press.(2006) 277-288
7. Euzenat, Jerome: Algebras of ontology alignment relations. *Proceedings of the 7th International Semantic Web Conference*, (pp. 387-402). Karlsruhe, Germany (2008)
8. Mitra, Prasenjit.: An Ontology Composition Algebra. *International Handbooks on Information System*, SpringerVerlag (2004) 93-117.
9. Mocan, A. C.: Formal Model for Ontology Mapping Creation. *Proceedings of the 5th International Semantic Web Conference*. Athens, USA: Springer (2006) 459-472
10. Richard, C.: *RDF 1.1 Concepts and Abstract Syntax*. Retrieved January 20, 2015, from <http://www.w3.org/TR/rdf11-concepts/> (2014)
11. F. William Lawvere and Stephen H. Schanuel: *Conceptual Mathematics, A first Introduction to Categories*. United Kingdom: Cambridge University Press (2009)
12. Awodey Steve: *Category Theory [Oxford Logic Guide]*. Carnegie Mellon University: Clarendon Press Oxford. (2006)
13. Micheal Barr and Charles Wells: *Category Theory for Computing Science*. McGill University (1998)
14. Patrick J. Hayes and Peter F. Patel-Schneider : RDF 1.1 Semantics. *W3C Recommendation* p. 4. Retrieved February 25, 2014 (2014)
15. David I. Spivak and Robert E. Kent. OLOGS: A Categorical framework for Knowledge Representation. *PLoS One* (2012) 31;7(1):e24274. Epub 2012 Jan 31.

16. Sowa, J. *Knowledge Representation-Logical, Philosophical and Computational Foundations*. Brooks/Cole. 2000.
17. Natalya F. Noy and Mark A. Musen. SMART: Automated Support for Ontology Merging and Alignment. In *Twelfth Banff Workshop on Knowledge Acquisition, Modeling, and Management* . 1999.
18. Deborah L. McGuinness, Richard Fikes, James P. Rice, and Steve Wilder. The Chimaera Ontology Environment. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence*. 2000.