Measuring inconsistencies propagation from change operation based on ontology partitioning

Mouhamadou Gaye
Numerical Analysis and Computer Laboratory – UFR Applied Sciences and Technology
Gaston Berger University, UGB
Saint-Louis, SENEGAL
gaye.mouhamadou@ugb.edu.sn

Ousmane Sall
Department of Computer Science - UFR Science and Technology
Thies University
Thies, SENEGAL
osall@univ-thies.sn

Mamadou Bousso
Department of Computerized Management Organisations - UFR Economic and Social Sciences
Thies University
Thies, SENEGAL
mbousso@univ-thies.sn

Moussa Lo
Numerical Analysis and Computer Laboratory – UFR Applied Sciences and Technology
Gaston Berger University, UGB
Saint-Louis, SENEGAL
moussa.lo@ugb.edu.sn

***Abstract*—Inconsistency measure is an activity related to the ontology evolution. Being a coherent entity, an ontology must change and a modification operation in ontology could generate inconsistencies in its other parts. It is then important to measure these inconsistencies and follow the impact propagation. In this paper, we propose an inconsistency measure of an ontological change and its propagation effects on the other entities of the ontology. The measure is based on the weight of the dependences between concepts in a community. Ontology is divided into communities which are a set of concepts that have preferential relations. To follow the impact propagation, we propose a process that uses the Change-and-Fix' approach to mark the impacted entities.**

***Keywords*—Inconsistency; Measure; Community; Evolution; Propagation; Ontology**

I. Introduction

Ontology evolution refers to the process of modifying ontology in response to a change in its conceptualization [6]. As a coherent entity, ontology may evolve and each change operation on components may bring inconsistency to other taxonomic and semantic components. It is then important to measure the degree or level of inconsistency of an entity change operation to one ontology, in order to define appropriate actions that will steer the system to a consistent state. An inconsistency measure quantifies the contribution of each axiom or element of a knowledge base in all inconsistencies produced in this base. It gives a schema of the inconsistency severity in the knowledge base. Many measurement approaches have been published. [7] proposes a method for inconsistencies reduction by splitting formulas while the approach proposed in [17] defines a degree of inconsistency of a DL-Lite ontology using a method called the "three-valueSd semantics". The algorithm proposed a PTIME complexity measure. Shapley values are the support of inconsistency measure proposed in [9] whose model is independent of any reasoning language. The approach in [5] presents a method for measuring the inconsistency based on the Dempster-Shafer theory of evidence. The algorithm calculates inconsistencies brought by each atom, each formula, each set of formulas and derives ontology inconsistency from these measurements. In [3] the proposed approach combines the Shannon entropy measure of the satisfiability concept and quantity of information provided after an ontology change operation. The weakness of the proposed method is that Shannon entropy doesn't clearly allow a comparison of ontology structural and semantic information before and after a change operation. We proposed in [13], a change modeling approach based on Hoare axiomatic semantics that allow satisfiability tests depending on different operations. However, we don't address the assessment of the impact on dependent entities.

In this paper, we propose an inconsistency measure of an ontological change and its propagation effects on ontology entities. The measure is based on weight dependencies between concepts in a community. Ontology is divided into syntactic communities, which are a set of concepts that have preferential relations. There are different works on partitioning large ontologies. Stuckenschmidt and Schlicht [14] or Grau and al. [8] decompose ontology into independent sub-blocks to facilitate maintenance, visualization, validation or reasoning. Noy and Musen [10] allow user to extract portions of ontologies centered on one or several concepts and specify relationships between them. Our decomposition method is based on the approach in [16] where concept hierarchies are used to extract. We propose a propagation process that marks the impacts flow of ripple effects resulting from changes. We apply our approach to the Food Ontology [18].

The rest of the article is organized as follows. We start by giving some basic notions about our model. In Section 3, we describe a community detection algorithm and process for entities dependency calculation. Section 4 and section 5 are respectively devoted to inconsistency measure definition and impact propagation description. A validation of the approach based on Food Ontology is given in section 6 and we conclude with a summary and outlook.

II. PRELIMINARIES

A. *Formal model of ontology*

Several models exist for ontology representation such as the lexical model. In this article, we use the lexical model formalized in [12]. A model of lexical ontology is defined as a set O = <S, L> where S is the structure and L the lexical level. Thus, the structure of an ontology O with which a lexicon is associated is the tuple:

$S = \{C, R, A, T, CAR_R, H^C, \sigma_R, \sigma_{CARR}, \sigma_A, \sigma_T\}$ where:

- C, A, T, CAR_R are respectively sets containing, the concepts of ontology, the relations of attribute, the types of attribute and characteristics of associative relations;
- $R \subseteq (C \times C)$ is associative relations set. It makes it possible to define the semantic types of relations connecting the concepts of ontology in (C x C);
- H^C hierarchy (taxonomy) of concepts: $H^C \subseteq (C \times C)$, $H^C(C_i, C_j)$ means that C_i is a sub-concept of C_j, for subsumption relations between ontology concepts;
- σ_R: $R \rightarrow C \times C$ is the signature of an associative relation. We will note σ_R (C_i, R_k, C_j) the signature of the associative relation R_k between the concepts C_i and C_j;
- σ_A: $A \rightarrow C \times T$ is the relation of attribute signature, T is composed of the simple types. It is noted as $\sigma_A(C_i, A_k, T_j)$ specifying the relation of attribute between a concept C_i and a A_k attribute having values of the T_j type;
- σ_T : $A \rightarrow T$ is the signature of the relation associating with an attribute A_k, the T_j type in the form $\sigma T(A_k, T_j)$ specifying that the A_k attribute is associated with values of the T_j type;
- σ_{CARR}: $R \rightarrow CAR_R$ is the relation specifying the characteristic of an associative relation. We will, thus, note an associative relation R_k transitive by signature $\sigma_{CARR}(R_k, Trans)$.

Example 1: Consider ontology O_1 on the auto mechanics defined as follows:

C = {Lorry, Vehicle, Engine, Box, Doors, Wheels, Cylinder, Petrol, Radiator, Water, Diesel}
R = {is_composed, is_formed, carry_away, turns, consumes, cools}
A = {costs}
T = {string}
S_{HC} = {H^c(Lorry, Vehicle), H^c(Diesel, Petrol)}
$S_{\sigma R}$ = { σ_R(Vehicle, is_composed, Box), σ_R(Vehicle, is_composed, Doors), σ_R(Vehicle, is_composed, Box), σ_R(Vehicle, is_composed, Engine), σ_R(Vehicle, is_composed, Wheels), σ_R(Engine, is_formed, Cylinder), σ_R(Engine, is_formed, Radiator), σ_R(Box, carry_away, Engine), σ_R(Engine, consumes, Petrol), σ_R(Engine, consumes, Water), σ_R(Water, cools, Radiator), σ_R(Engine, turns, Wheels)}
$S_{\sigma A}$ = {σ_A(Diesel, costs, « 1.05 euro »)}

TABLE I. BASIC AND GENERAL ASSERTIONS

Id	Assertion	Signification
Positive Assertions	$+C_i$	$\exists (C_i \in C)$
	$+R_i$	$\exists (R_i \in R)$
	$+A_i$	$\exists (A_i \in A)$
	$+T_i$	$\exists (T_i \in T)$
	$+CAR_{Ri}$	$\exists (CAR_{Ri} \in CAR_R)$
	$+H^C(C_i, C_j)$	$\exists (Ci \in C \wedge C_j \in C) / H^C(C_i, C_j)$
	$+\sigma_R(C_i, R_k, C_j)$	$\exists (C_i \in C, C_j \in C \wedge R_k \in R) / \sigma_R(C_i, R_k, C_j)$
	$+\sigma_{CARR}(C_i, CAR_{Ri})$	$\exists (C_i \in C \wedge CAR_{Ri} \in CAR_R) / \sigma_{CARR}(C_i, CAR_{Ri})$
	$+\sigma_A(C_i, A_j, T_k)$	$\exists (C_i \in C, A_j \in A \wedge T_k \in T) / \sigma_A(C_i, A_j, T_k)$
Negative Assertions	$-C_i$	$\neg (C_i \in C)$
	$-R_i$	$\neg (R_i \in R)$
	$-A_i$	$\neg (A_i \in A)$
	$-T_i$	$\neg (T_i \in T)$
	$-CAR_{Ri}$	$\neg (CAR_{Ri} \in CAR_R)$
	$-H^C(C_i, C_j)$	$\forall (C_i \in C) \wedge \forall (C_j \in C) : \neg H^C(C_i, C_j)$
	$-\sigma_R(C_i, R_k, C_j)$	$\forall (C_i \in C) \wedge \forall (C_j \in C) \wedge \forall (R_k \in R) : \neg \sigma_R(C_i, R_k, C_j)$
	$-\sigma_{CARR}(C_i, CAR_{Ri})$	$\forall (C_i \in C) \wedge \forall (CAR_{Ri} \in CAR_R) : \neg \sigma_{CARR}(C_i, CAR_{Ri})$
	$-\sigma_A(C_i, A_j, T_k)$	$\forall (C_i \in C) \wedge \forall (A_j \in A) \wedge \forall (T_k \in T) : \neg \sigma_A(C_i, A_j, T_k)$
	$-H^C(*, C_i)$	$\forall (C_k \in C) : \neg H^C(C_k, C_i)$
	$-H^C(C_i, *)$	$\forall (C_k \in C) : \neg H^C(C_i, C_k)$
	$-\sigma_{CARR}(R_i, *)$	$\forall (CAR_{Ri} \in CAR_R) : \neg \sigma_{CARR}(Ri, CAR_{Ri})$
	$-\sigma_A(*, A_i, T_j)$	$\forall (C_k \in C) \wedge \forall (A_i \in A) \wedge \forall (T_j \in T) : \neg \sigma_A(C_k, A_i, T_j)$
	$-\sigma_A(*, A_i, *)$	$\forall (C_k \in C) \wedge \forall (T_j \in T) : \neg \sigma_A(C_k, A_i, T_j)$
	$-\sigma_A(C_k, *, *)$	$\forall (A_i \in A) \wedge \forall (T_j \in T) : \neg \sigma_A(C_k, A_i, T_j)$
	$-\sigma_A(*, *, T_j)$	$\forall (C_k \in C) \wedge \forall (A_j \in A): \neg \sigma_A(C_k, A_i, T_j)$
	$-\sigma_R(C_i, *, *)$	$\forall (C_k \in C) \wedge \forall (R_j \in R): \neg \sigma_R(C_i, R_i, C_k)$
	$-\sigma_R(*, R_k, *)$	$\forall (C_i \in C) \wedge \forall (C_j \in C) : \neg \sigma_R(C_i, R_k, C_j)$

III. IDENTIFYING COMMUNITIES

The partitioning method is inspired by approach proposed in [16]. This approach enables a decomposition of an ontology based on the structure of the hierarchy of concepts. Our method uses subsumption and associative relationships between nodes in the ontological graph in the decomposition criteria.

A. *Dependency graph*

We start by creating a weighted graph.

Definition 1 ontological weighted graph

An ontological weighted graph is a tuple G = (E, Γ, W) where E is a set of concepts, Γ an application from E to P(E), where P(E) contains all the set included in E and W a weighting function determining the relationship between an element in E and an element in P (E).

Ontology concepts are linked by subsumption relationships H^c, associative relationships σR and attribute relationships σA. For each type of relationship between two concepts C_i and C_j, we assign a weight p_{ij} according to the direction of propagation flow impact:

- if $H^c(C_i, C_j)$ then $p_{ji} = 1$ and $p_{ij} = 0$;
- if $\sigma_R(C_i, R_k, C_j)$ then $p_{ji} = 1$ and $p_{ij} = 0$.

We don't address in our approach attribute relationships and these values are justified by the results of our work in [12]. We considered that three relationships can spread impacts to the target entities. The subsumption relationship $H^c(C_i, C_j)$ indicates that any change on C_j can impact C_i. In the same paper, it was established that for associative relationships $\sigma_R(C_i, R_k, C_j)$, change on C_j can impact C_i except for the equivalence relationships. For the attribute relationship $\sigma_A(C_i, A_k, T_j)$, a change in the attribute A_k may have consequences for the concept C_i that uses it.

Définition 2 Weight of a dependency

Let G = (E, Γ, W) be an ontological weighted graph, C_i and C_j two concepts of E. We define the weight of the dependence between the concepts C_i and C_j as follows:

$$\mathbf{w(Ci, Cj)} = \frac{pij + pji}{\sum_{k=1}^{N} pik + pki} \qquad (1)$$

N is the number of concepts to which C_i is connected in G.

This weight will be used in the algorithm for communities' detection on ontology.

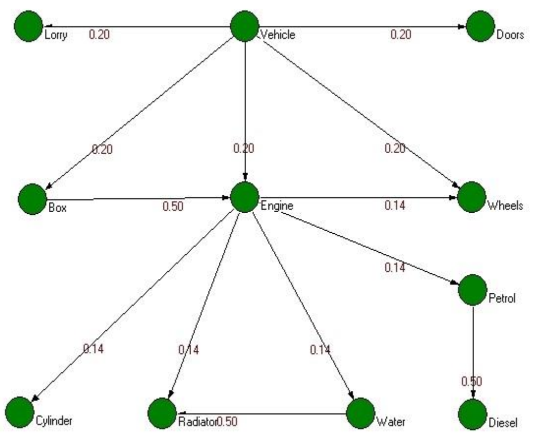

Fig. 1. Ontological weighted graph of the ontology in example 1

B. *Partitioning graph*

Ontology identification communities' can be seen as a problem of building concepts clusters. The particularity is that a concept can belong to one or more communities. Managing evolution of large ontologies is not an easy task. This subdivision into communities makes managing very large ontologies for example in medicine or biology easier, particularly the inconsistency measure and the impact propagation.

Définition 3 Community

A community is a set of concepts that share more intra properties inside more than outside of the community.

We use in our approach the Line Islands algorithm defined in [1] to break ontology into communities. This algorithm determines the maximum of lines separation in an ontological graph. The number of lines separation is variable, depending on the size of the ontology.

Définition 4 Edge island

A set of nodes V is an edge island if:

- It is a singleton or;
- The subgraph corresponding is a connected graph such that:

$$\max_{ci \in V \wedge cj \notin V} w(Ci, Cj) \le \min_{ck, cl \in V} w(Ck, Cl) \qquad (2)$$

Edge island V ⊆ G is regular edge island, if stronger condition holds:

$$\max_{ci \in V \wedge cj \notin V} w(Ci, Cj) < \min_{ck, cl \in V} w(Ck, Cl)$$

Algorithm1 Partitioning ontology

Input: G = (E, Γ, W) a ontological weighted graph, maxCties maximum number of communities

Output: counter the number of communities obtained.

```
1 min = 1
2 max = |E| - 1
3 islands = {{v} : v ∈ E}
4 for all i ∈ islands do i.port = 0 (vertex with the smallest weight)
5 sort E in decreasing order according to the weight w
6 for all e(u, v) ∈ G do
7       i1 = island ∈ islands : u ∈ island
8       i2 = island ∈ islands : v ∈ island
9       if i1 ≠ i2 then
10            island  = new Island()
11            island.port = e
12            island.subisland1 = i1
13            island.subisland2 = i2
14            islands  = islands ∪{island}\{i1,i2}
17      endif
18 endfor
19 candidates = ∅
20 while islands = ∅ do
21      select island ∈ subislands
```

```
22      subislands = subislands \{island}
23      if |island| < min then
24              delete island
25      else if |island| > max then
27              islands = islands ∪ {island.subisland1, island.subisland2}
28              delete island
29          else
30              candidates = candidates ∪ {island}
31          endif
32      endif
33 endwhile
34 for all module ∈ candidates do
35      expand(module, maxCties)
36      partition(maxCties, module, counter)
37 endfor
```

The proposed algorithm like those which make a depth search in a graph is an O(n+m) complexity with n the number of node and m the number of arcs., so it's linear.

Note that with this algorithm a concept can't belong in more than one community and an isolated concept doesn't form a community. Isolated concepts are linked to communities with which they are closest. This is calculated using dependency weight.

IV. INCONSISTENCIES MEASUREMENT

We start by giving basic change operations that are listed in the following table. The used assertions are defined in table 1.

TABLE II. EXAMPLES OF BASIC OPERATIONS

Id	Basic operations	Pre-condition	Invariant	Post-condition
1	CreateConcept(C_i)	-C_i	-H^C (*,C_i) -σ_R(*, *, C_i)	+C_i
2	DeleteConcept(C_i)	+ C_i	-H^C (*,C_i) -σ_R(*, *, C_i)	- C_i
3	CreateAssociative Relation(R_i)	-R_i -σ_{CARR}(R_i,*)	-σ_R(*,R_i, *)	+ R_i +σ_{CARR}(R_i, $_{CARRi}$)
4	DeleteAssociative Relation(R_i)	+ R_i	-σ_{CARR}(R_i,*) -σR(*,R_i, *)	- R_i
5	CreateProperty(A_i,T_i)	-A_i	-σ_A(*,A_i,T_i) -σ_T(A_i, *) and + T_i	+ A_i +σ_T(Ai, T_j)
6	DeleteProperty(Ai)	+Ai	-σ_A(*,A_i,T_i) -σ_T(A_k,*) and + T_i	- A_i

Definition 5 Free Subset

Let K be a knowledge base. We define Free(K) as the set contains the formulae in K that are not involved in any inconsistency.

The inconsistency measure is based on weight of the dependencies in a community. Modification operations concerned are simple changes such as creating and deleting entities. In [15], Stojanovic shows that any complex change can be transform into atomic changes and so we don't need to address complex changes. We specified in [12] that each operation is associated with a whole of assertions declined in three possible cases:

- If the pre-condition and the invariant are checked, then the operation can be carried out without propagation of impacts;
- If the pre-condition is not checked, then the operation is not checked and there is no impact on the ontological components;
- If the pre-condition is checked and that the invariant is not then checked the operation is carried out and there is an impact propagation process that we propose to measure.

A modification operation is modeled like a triplet Δ=<Op, Args, Assert> representing the operation, its arguments and Assert =< Pre, Inv, Post> for pre-conditions, invariant conditions, post-conditions as in table 2. The inconsistency measure of the operation Δ consists in measuring the base inconsistency K that contains the negation of the invariant set Inv defined in table 2.

A. Measuring inconsistency in the community

Definition 6

Let be Op(x) a modification operation such as CreateEntity(x) or DeleteEntity(x). The inconsistency measure of an entity modification Op(x) in a community C can be defined as follows:

$$\mathrm{Ic}(\mathrm{Op}(\mathrm{x})) = \mathrm{Ic}(\mathrm{K} = \neg\mathrm{Inv})\frac{\sum_{e\in K} w(x,e)}{\sum_{\mathrm{u,v}\in C} \mathrm{w(u,v)}} \tag{3}$$

where C represents a community, Inv the invariant of the operation Op and K the set that contains the negation of the invariant set Inv.

Proposition

Ic is a measure in K.

Proof:

We must prove the three assertions:

1. Consistency: Ic(K) = 0 if K is consistent.
2. Monotony: If K$\subseteq$K', then Ic(K) $\leq$ I(K').
3. Free Formula Independence: For all $\alpha \in$ Free(K); I(K) = Ic(K\{ α }).

Let be K the invariant negation of a modification operation Op(x).

1. Ic(Op(x)) = Ic(K) = 0 $\Leftrightarrow$ $\forall$ u $\in$ K, w(x, u) = 0

   $\Leftrightarrow \dfrac{pij + pji}{\sum_{k=1}^{N} pik + pki} = 0$ with C_i = x and C_j = u

   $\Leftrightarrow$ $pij + pji$ = 0 $\Leftrightarrow$ x has no defined relation in K

   $\Leftrightarrow$ K is consistent.

2. Let $K \subseteq K'$, then :

$$Ic(K') = \frac{\sum_{e\in K'} w(x,e)}{\sum_{u,v\in C} w(u,v)}$$

$$= \frac{\sum_{e\in K} w(x,e) + \sum_{e\in K', e\notin K} w(x,e)}{\sum_{u,v\in C} w(u,v)}$$

$$= \frac{\sum_{e\in K} w(x,e)}{\sum_{u,v\in C} w(u,v)} + \frac{\sum_{e\in K', e\notin K} w(x,e)}{\sum_{u,v\in C} w(u,v)}$$

$$= I(K) + \frac{\sum_{e\in K', e\notin K} w(x,e)}{\sum_{u,v\in C} w(u,v)} \geq I(K) \text{ in the fact}$$

that $\sum_{e\in K', e\notin K} w(x,e) \leq \sum_{u,v\in C} w(u,v)$

3. Let be $\alpha \in Free(K)$, then $Ic(\{\alpha\}) = 0$ from the consistency.
Or $Ic(K) = Ic(K \backslash \{\alpha\}) + Ic(\{\alpha\})$, therefore
$Ic(K) = Ic(K \backslash \{\alpha\})$.

Example 2: Suppose that a deletion operation concept Box in the ontology O_1 in example 1 is done. The concept Box is in the community C = {Lorry, Vehicle, Doors, Box, Engine, Wheels} that shows the following figure:

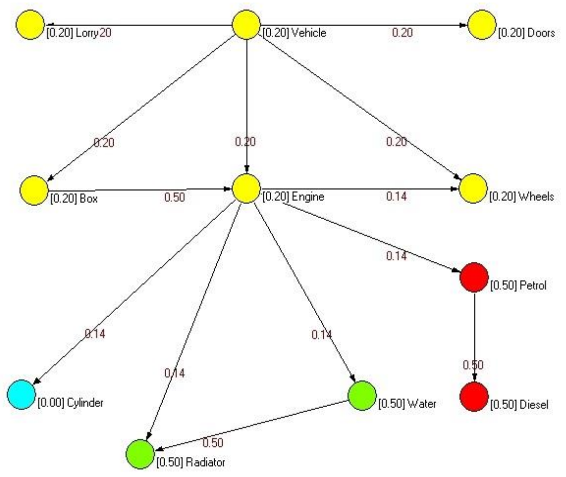

Fig. 2. Communities of the ontology in example 1

And then the assertions are:

- Pre-condition = {+Box};
- Post-condition = {- Box};
- Invariant = $\{-H^C(*,Box), -H^C(Box, *), -\sigma_A(Box, *, *), -\sigma_R(Box, *, *)\}$.

The negation Invariant is :

$K = \{+H^C(*,Box), +\sigma_A(Box, *, *), +\sigma_R(Box, *, *)\}$

$K = \{+H^C(C_i, Box), +\sigma_A(Box, R_k, C_j), +\sigma_R(Box, A_k, C_l)\}$

$K = \{\sigma_R(Vehicle, is\_composed, Box), \sigma_R(Box, carry\_away, Engine)\}$

$$Ic(DeleteConcept(Box)) = \frac{\sum_{e\in K} w(x,e)}{\sum_{u,v\in C} w(u,v)}$$

$$Ic(DeleteConcept(Box)) = \frac{w(Box,Vehicle) + w(Box,Engine)}{\sum_{u,v\in C} w(u,v)}$$

or $\sum_{u,v\in C} w(u,v) = 1.64$

thus

$$Ic(DeleteConcept(Box)) = \frac{0 + 0.50}{1.64} = 0.30$$

B. *Measuring inconsistency in the ontology*

We consider here the inconsistency of a modification operation in the ontology.

Definition 7 Inter-community relationship

Two communities C and C' are connected if there exist a $C_i \in C$ and $C_j \in C'$ such as $w(C_i, C_j) \neq 0$.

Definition 8

Let be Op(x) a modification operation such as CreateEntity(x) or DeleteEntity(x). The impact inconsistency measure of an entity modification Op(x) in the ontology O can be defined as follows:

$$I_o(Op(x)) = \frac{I_C(Op(x)) * Nb\_rel(C)}{N} \qquad (4)$$

where O designs the ontology, C represents the community that contains the entity x, Nb_rel(C) number of communities that C is connected and N number of communities in O.

Example 3: The deletion operation concept Box in the ontology O_1 in example 1 gives:

$$I_o(DeleteConcept(Box)) = \frac{0.30 * 2}{3} = 0.2$$

V. CHANGES PROPAGATION

In this section, we propose an algorithm that takes an ontology and a change operation as inputs and gives as output the inconsistencies propagation path in the ontology. All concepts that take account in these inconsistencies are marked.

This algorithm is based on 'Change-and-Fix' approach proposed by Rajlich [11] and Deruelle [4] for change impact analysis.

Algorithm 2 Change propagation
Input: O an ontology, Op a modification operation
Output: P a set of marked concepts

```
1 ExecuteOperation(Op)
2 Inv = Op.invariant
3 P = ∅
3 for all $Cond_i \in Inv$ do
4         if (false($Cond_i$)) then
5                 mark($Cond_i$)
6                 $P = P \cup Cond_i$
7         endif
8 endfor
```

The following table shows how to mark a condition.

TABLE III. ASSERTIONS FOR MARKING

Id	Assertion	Signification
1	markConcept(C_i)	$\forall(C_j \in C)$ if $H^C(C_j, C_i)$ then markRelation($H^C(C_j, C_j)$) $\forall(C_j \in C)$ and $\forall$ ($R_k \in R$) if $\sigma R(C_j, R_k, C_i)$ then markRelation($\sigma R(C_j, R_k, C_j)$)
2	markRelation(R_k)	$\forall(C_j \in C)$ and $\forall(C_j \in C)$ if $\sigma R(C_j, R_k, C_j)$ then markRelation($\sigma R(C_j, R_k, C_j)$)
3	markProperty(A_k)	$\forall(C_j \in C)$ if $\sigma A(C_j, A_k, T_j)$ then markRelation($\sigma A(C_i, A_k, T_j)$)
4	markRelation($H^C(C_j, C_i)$)	if (C_j not marked) then markConcept(C_j)
5	markRelation(σR (Ci, R_k, Cj))	if (C_j not marked) then markConcept(C_j)
6	markRelation(σA (C_j, A_k, T_j))	if (A_k not marked) then markConcept(A_k)
7	markRelation($H^C(*, C_j)$)	$\forall(C_k \in C)$ if ($H^C(C_k, C_j)$) and (C_k not marked) then markConcept(C_k)
8	markRelation($\sigma A(*, A_i, T_j)$)	$\forall C_k \in C$, if ($\sigma A(C_k, A_i, T_j)$) and (C_k not marked) then markConcept(C_k)
9	markRelation($\sigma A(*, *, T_j)$)	$\forall C_k \in C$, $\forall A_i \in A$ if ($\sigma A(C_k, A_i, T_j)$) then if (C_k not marked) then markRelation($\sigma A(C_k, A_i, T_j)$) if (A_i not marked) then markRelation($\sigma T(A_i, T_j)$)
10	markRelation($\sigma R(*, *, C_k)$)	$\forall R_j \in C$, $\forall C_j \in C$: if ($\sigma R(C_j, R_j, C_k)$) then if (C_i not marked) then markConcept(C_j)
11	markRelation($\sigma R(*, R_k, *)$)	$\forall C_j \in C$, $\forall C_j \in C$: if ($\sigma R(C_j, R_k, C_j)$) then if (C_j not marked) then markConcept(C_j)

VI. VALIDATION

We implemented the approach first on the Food Ontology. This ontology describes the different types of food that exist. It contains 63 concepts. We used the Pajet tool [2] to partition ontology and view its different communities. We first create the .net file that Pajet takes as input by transforming the .owl file. We construct the adjacency weighted matrix of the ontological graph m (m[i][j] contains $w(C_i, C_j)$ defined in (1)). Thus, we obtained 10 communities that are shown in the following figure.

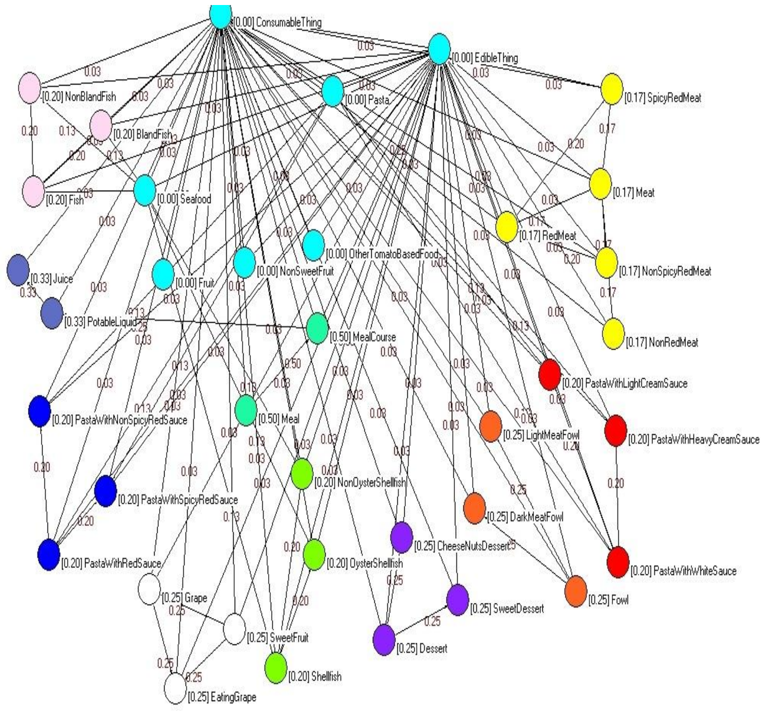

Fig. 3. Communities obtained with food ontology

To view the change propagation process, we used the adjacency matrix M constructed as follows.

Let be C_i and C_j two concepts of G, then:

- if $H^c(C_i, C_j)$ then $M_{ji} = 1$ and $M_{ij} = 0$;
- if $\sigma_R(C_i, R_k, C_j)$ then $M_{ji} = 1$ and $M_{ij} = 0$.

When a modification operation occurredon a concept C_i, the marking process determines all concepts C_j such as $M_{ij} = 1$. This process is repeated until there is no concept to be marked. In figure 4 we show the change propagation resulting from the deletion of the concept Box in ontology O_1 proposed in example 1. The concepts with value 1 are marked.

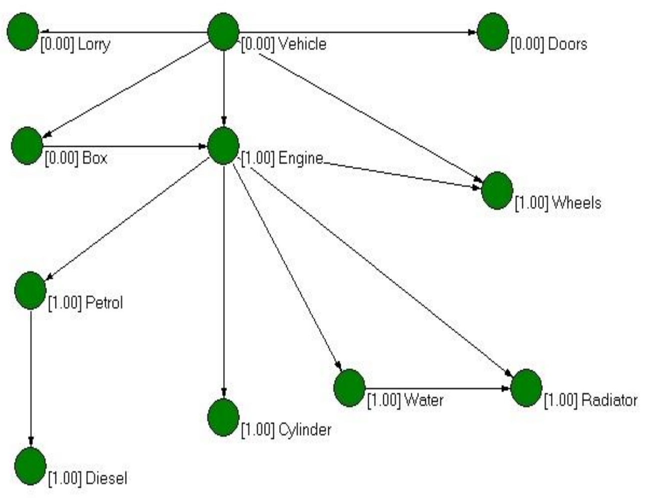

Fig. 4. Change propagation resulting for the deletion of the concept Box

VII. CONCLUSION

In this paper, we present an inconsistency measure of an ontology change operation and its propagation effects on ontology entities. The measure is based on dependencies weight between concepts in communities. Ontology is divided into syntactic communities, which are a set of concepts that have preferential relations. The communities' identification is guided by subsumption and associative relationships between nodes in the ontological graph.

In future work, we plan to complete the development of this framework on large ontology like Gene Ontology and propose algorithms for planning inconsistency resolution based on markovian methods.

REFERENCES

[1] Batagelj, V.: Analysis of large networks-islands. Presented at Dagstuhl seminar 03361: Algorithmic Aspects of Large and Complex Networks (2003)

[2] Batagelj, Vladimir and Andrej Mrvar. 1998. "PAJEK -- Program for large network analysis." Connections, 21:47-57.

[3] Bousso, M. Sall, O., Thiam, M., Lo, M., Touré, E. H. B.: Ontology Change Estimation Based on Axiomatic Semantic and Entropy Measure. IEEE Conference- Track Signal and Image Technology of the 8th International Conference on Signal-Image Technology and Internet-Based Systems (SITIS 2012) 25-29, November (2012). J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., Vol. 2. Oxford: Clarendon, 1892, 68–73. [3x] L. Dongmei, L. Youfang, H. Houkuan, H. Shudong, W. Jianxin, "Dempster-Shafer Inconsistency Values", In Chinese Journal of Electronics Vol. 23, No. 2, Apr. 2014.

[4] Deruelle, L.: Analyse d'impact de l'évolution des applications distribuées multi-langages et à bases de données hétérogènes. Thèse de doctorat. Université du Littoral Côte d'Opale (2001)

[5] Dongmei, L., Youfang, L., Houkuan, H., Shudong, H., Jianxin, W.: Dempster-Shafer Inconsistency Values. In Chinese Journal of Electronics Vol. 23, No. 2 (2014)

[6] Flouris, G., Plexousakis, D., Antoniou, G.: A Classification of Ontology Change. In Proceedings of the 3rd Italian Semantic Web Workshop, Semantic Web Applications and Perspectives (SWAP) (2006).

[7] Grant, A. Hunter, "Measuring Consistency Gain and Information Loss in Stepwise Inconsistency Resolution", In ECSQARU, pages 362–373, 2011

[8] Grau, B. C., B. Parsia, E. Sirin, et A. Kalyanpur (2005). Automatic partitioning of owl ontologies using e-connections. In DL2005, Proceedings of 18th International Workshop on Description Logics, Edinburgh, UK.

[9] Hunter, A.: On the Measure of Conflicts: Shapley Inconsistency Values, 2010.

[10] Noy, N. F. et M. A. Musen (2000). PROMPT: Algorithm and tool for automated ontology merging and alignment. In AAAI/IAAI, pp. 450–455.

[11] Rajlich, V(1997), A Model for Change Propagation Based on Graph Rewriting. In Proceedings of the international Conference on Software Maintenance (October 01 - 03, 1997). ICSM. IEEE Computer Society, Washington, DC, 84-91.

[12] Sall, O., Thiam, M., Lo, M., Basson, H.: A model for ripple effects analysis of cascading problems in ontology evolution. Int. J. Metadata, Semantics and Ontologies, Vol. 7, No. 3 (2012).

[13] O. Sall, M. Thiam, M. Bousso, M. Lo "Using Hoare's Axiomatic Semantics For Checking Satisfiability of Ontology Change Operations", In IEEE Conference - 8th International Conference on Information Science and Digital Content Technology ICIDT'2012 (5th ICIS) - June 26 to 28, 2012 in Jeju Island, Republic of Korea, p. 61-66, Vol. 1, ISBN: 978-1-4673-1288-2.

[14] A. Schlicht and H. Stuckenschmidt. Criteria-based partitioning of large ontologies. In Proceedings of the International Conference on Knowledge Capture (K-CAP), 2007.Poster Contribution

[15] Stojanovic, L.: Methods and Tools for Ontology Evolutio. University of Karlsruhe (2004).

[16] H. Stuckenschmidt and M. Klein. Structure-based partitioning of large concept hierarchies. In S. A. McIlraith, D. Plexousakis, and F. van Harmelen, editors, Proceedings of the Third International Semantic Web Conference(ISWC2004), pages 289–303, Hiroshima, Japan, nov 2004.

[17] L. Zhou, H. Huang, G. Qi, Y. Ma, Z. Huang, Y. Qu, "Measuring Inconsistency in DL-Lite Ontologies", IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology – Workshops, 2009.

[18] Food Ontology. http://www.w3.org/TR/2003/PR-owl-guide-20031215/food