Define Hybrid Class Resolving Disjointness Due to Subsumption

http://ontologydesignpatterns.org/wiki/Submissions:Define_Hybrid_Class_Resolving_Disjointness_due_to_Subsumption

Rim Djedidi[1] and Marie-Aude Aufaure[2]

[1] Computer Science Department, Supélec Campus de Gif
Plateau du Moulon – 3, rue Joliot Curie – 91192 Gif sur Yvette Cedex, France
rim.djedidi@supelec.fr
[2] MAS Laboratory, SAP Business Object Chair –Centrale Paris
Grande Voie des Vignes, F-92295 Châtenay-Malabry Cedex, France
marie-aude.aufaure@ecp.fr

1 Introduction

The pattern "*Define Hybrid Class Resolving Disjointness due to Subsumption*" is proposed as a Logical Ontology Design Pattern (Logical OP) solving a problem of disjointness inconsistency caused by a subsumption relation. Further away from solving design problems where the primitives of the representation language do not directly support certain logical constructs, this pattern helps resolving a logical inconsistency triggered by a situation of disjoint classes subsuming a common sub-class. The solution presented by the pattern resolves the inconsistency while preserving existing knowledge, i.e. a resolution alternative avoiding axiom deletion.

2 Pattern

In this section, we specify the problem that the pattern deals with and the requirements covered by it; we detail the description of the solution given by this pattern and the consequences of its application; and we illustrate the pattern by an example problem and its corresponding solution.

2.1 Problem

The pattern "*Define Hybrid Class Resolving Disjointness due to Subsumption*" is proposed to solve a problem of disjointness inconsistency caused by a subsumption relation. When we need to define – for some modeling issues related to domain of interest – a class as a sub-class of two disjoint classes, a disjointness inconsistency is caused.

The problem can be illustrated by the following scenario: let's consider a class *Sub_Class* defined as a sub-class of a class *Disjoint_Class 2*; and a class

Disjoint_Class 1 disjoint with the *Disjoint_Class 2* (Fig. 1). If we need to add a sub-class relation between the *Sub_Class* and the *Disjoint_Class 1*, this generates a disjointness inconsistency:

- If the extension of the *Sub_Class* contains individuals instantiating this sub-class, the logical inconsistency will be extended to the knowledge base;
- If the *Sub_Class* is not instantiated to individuals, it will be diagnosed as an unsatisfiable class.

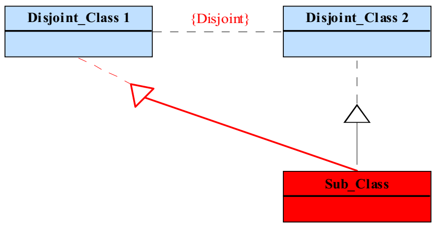

Fig. 1. Graphical illustration of the problem the pattern deals with.

To solve this inconsistency, one can think about deleting the disjointness axiom. However, this can alter the semantics expressed in the ontology, and negatively affect consistency checking and automatic evaluation of existing individuals as explained in [1].

This pattern tackles the questions of how to resolve the inconsistency caused by such kind of subsumption while preserving existing knowledge.

Intent The purpose of this pattern is to support the semantics of a subsumption defined under two disjoint classes and resolve the resulting inconsistency.

Covered Requirements The pattern solves a problem of disjointness inconsistency caused by a subsumption relation without deleting the disjointness axiom so that existing knowledge can be preserved.

2.2 Solution

The pattern resolves a disjointness inconsistency –due to a subsumption–by defining a *Hybrid Class* based on the definition of disjoint classes implicated in the inconsistency; and redistributing correctly sub-class relations between the sub-class, the hybrid class, and the most specific common super-class of the disjoint classes implicated. The definition of the *Hybrid Class* is the union (OR) of the definitions of the disjoint classes.

The application of the solution can be described by the following process (Fig. 2):

1. The pattern defines a *Hybrid Class* as a union of the definitions of the disjoint classes implicated in the inconsistency to be resolved;
2. The pattern defines a subsumption between the most specific common super-class of the disjoint classes implicated in the inconsistency, and the *Hybrid Class* created;

3. The pattern defines a subsumption between the *Hybrid Class* and the sub-class involved in the inconsistency.

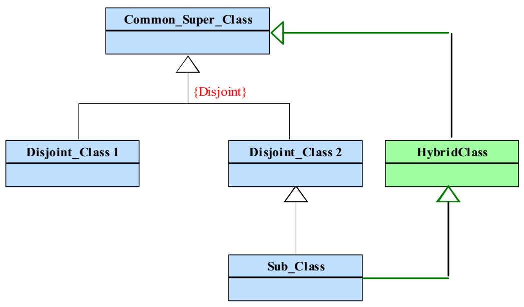

Fig. 2. Graphical representation of the proposed pattern.

Consequences The application of the pattern resolves the disjointness inconsistency (even if the involved sub-class is instantiated by individuals) and preserves existing knowledge. As a Logical OP, this pattern is independent from a specific domain of interest. However, it depends on the expressivity of the logical formalism used for the representation of the ontology. Therefore, the language of the targeted ontology should allow expressing class union.

2.3 Example

To explain pattern application, we present in this section, an example problem and its corresponding solution according to the pattern.

Example Problem Let's consider the OWL ontology O defined by the following axioms:

{Animal ⊑ Fauna-Flora, Plant ⊑ Fauna-Flora, Carnivorous-Plant ⊑ Plant, Plant ⊑ ¬Animal}

If we apply a change to the ontology defining *Carnivorous-Plant* class as a sub-class of the class *Animal*, we cause a disjointness inconsistency. The proposed pattern resolves this kind of inconsistency.

Example Solution The application of the pattern to resolve the example above is performed as follow:

1. The pattern defines a class *Animal_Plant* as a union of the definitions of the disjoint classes *Animal* and *Plant*;
2. The pattern defines a subsumption between the most specific common super-class of the disjoint classes *Fauna-Flora* and the hybrid class created *Animal_Plant*;

3. The pattern defines a subsumption between the defined hybrid class *Animal_Plant* and the sub-class *Carnivorous-Plant* involved in the inconsistency.

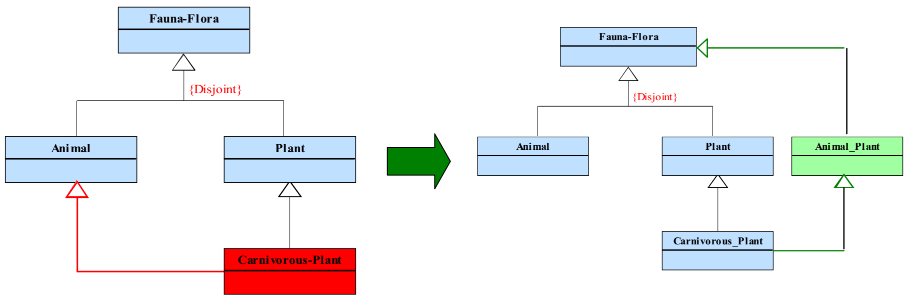

Fig. 3. Illustration of an example of problem and its corresponding solution.

3 Pattern Usage

The proposed – Logical OP – pattern "*Define Hybrid Class Resolving Disjointness due to Subsumption*" is applied as an *Alternative Resolution Pattern* in an ontology evolution approach ***ONTO-EVO^{A}L***, guided by *Change Management Patterns* (CMP) [2]. CMP patterns drive and control the change management process at three key phases: change specification, change analysis, and change resolution, by modeling three categories of patterns: *Change Patterns* classifying types of changes, *Inconsistency Patterns* classifying types of logical inconsistencies, and *Alternative Patterns* classifying types of inconsistency resolution alternatives.

4 Summary and Future Work

The purpose of this pattern is to support the semantics of a subsumption defined under two disjoint classes and resolve the resulting inconsistency without removing existing knowledge. This pattern can be extended and adapted to resolve disjointness inconsistency due to instantiation.

References

1. Völker, J., Vrandecic, D., Sure, Y., Hotho, A.: Learning Disjointness. In F., Enrico, K., Michael, May. Wolfgang (Eds.). Proceedings of the 4th European Semantic Web Conference, ESWC 2007. LNCS: Vol. 4519 pp: 175-189. (2007)

2. Djedidi, R., Aufaure, M-A.: Ontology Change Management. In: A. Paschke, H. Weigand, W. Behrendt, K. Tochtermann, T. Pellegrini (Eds.), I-Semantics 2009, Proceedings of I-KNOW '09 and I-SEMANTICS '09, ISBN 978-3-85125-060-2, pp. 611--621, Verlag der Technischen Universitt Graz. (2009).