OnlynessIsLoneliness (OIL)

http://ontologydesignpatterns.org/wiki/Submissions:OnlynessIsLoneliness_(OIL)

Oscar Corcho[1] and Catherine Roussey[23]

[1] Ontology Engineering Group, Departamento de Inteligencia Artificial, Universidad Politécnica de Madrid, Spain
ocorcho@fi.upm.es,
WWW home page:
http://www.dia.fi.upm.es/index.php?page=oscar-corcho&hl=en_US
[2] Cemagref, 24 Av. des Landais, BP 50085, 63172 Aubiére, France
catherine.roussey@cemagref.fr,
WWW home page: http://www.cemagref.fr/
[3] Université de Lyon, CNRS, Université Lyon 1, LIRIS UMR5205, Villeurbanne, France
catherine.roussey@liris.cnrs.fr,
WWW home page: http://liris.cnrs.fr/membres?idn=croussey

1 Introduction

Our work is based on the debugging process of real ontologies that have been developed by domain experts, who are not necessarily too familiar with DL, and hence can misuse DL constructors and misunderstand the semantics of some OWL expressions, leading to unwanted unsatisfiable classes. Our patterns were first found during the debugging process of a medium-sized OWL ontology (165 classes) developed by a domain expert in the area of hydrology [9]. The first version of this ontology had a total of 114 unsatisfiable classes. The information provided by the debugging systems used ([3], [5]) on (root) unsatisfiable classes was not easily understandable by domain experts to find the reasons for their unsatisfiability. And in several occasions during the debugging process the generation of justifications for unsatisfiability took several hours, what made these tools hard to use, confirming the results described in [8]. Using this debugging process and several other real ontologies debugging one, we found out that in several occasions domain experts were just changing axioms from the original ontology in a somehow random manner, even changing the intended meaning of the definitions instead of correcting errors in their formalisations.

We have identified a set of patterns that are commonly used by domain experts in their DL formalisations and OWL implementations, and that normally result in unsatisfiable classes or modelling errors ([1], [7]). Thus they are antipatterns. [6] define antipatterns as patterns that appear obvious but are ineffective or far from optimal in practice, representing worst practice about how to structure and build software. We also have made an effort to identify common alternatives for providing solutions to them, so that they can be used by domain experts to debug their ontologies.

All these antipatterns come from a misuse and misunderstanding of DL expressions by ontology developers. Thus they are all Logical AntiPatterns (LAP): they are independent from a specific domain of interest, but dependent on the expressivity of the logical formalism used for the representation.

2 Pattern

2.1 Problem

The ontology developer created a universal restriction to say that C_1 instances can only be linked with property R to C_2 instances. Next, a new universal restriction is added saying that C_1 instances can only be linked with R to C_3 instances, with C_2 and C_3 disjoint. Figure 1 illustrates this problem: grey squares represent instances of $C_2 \sqcap C_3$ that cannot exist. In general, this is because the ontology developer forgot the previous axiom in the same class or in any of the parent classes.

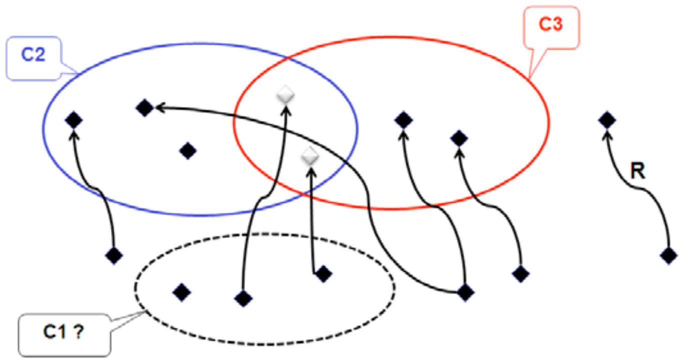

Fig. 1. A graphical representation of OIL antipattern.

$C_1 \sqsubseteq \forall R.(C_2); C_1 \sqsubseteq \forall R.(C_3); Disj(C_2, C_3);$ [4]

Notice that to be detectable, R property must have at least a value, normally specified as a (minimum) cardinality restriction for that class, or with existential restrictions.

Covers Requirements When this antipattern appears during the debugging process, you have to first explain to the domain expert the meaning of this formalisation using a schema like the one of the Figure 1. Then you could ask

[4] This does not mean that the ontology developer has explicitly expressed that C_2 and C_3 are disjoint, but that these two concepts are determined as disjoint from each other by a reasoner. We use this notation as a shorthand for $C_2 \sqcap C_3 \sqsubseteq \perp$.

him some questions to find out where is the problem. For example, you could ask:

- Should C_1 be linked with the R property to C_2?
- Should C_1 be linked with the R property to C_3?
- Does C_1 have to be linked only to C_2 with the R property?
- Does C_1 have to be linked only to C_3 with the R property?
- Are you sure that C_2 and C_3 are disjoint?

2.2 Solution

If it makes sense, we propose the domain expert to transform the two universal restrictions into only one that refers to the logical disjunction of C_2 and C_3. Another alternative solution, which is used by most part of automatic debugging tool is to remove one of the axioms.

$\cancel{C_1 \sqsubseteq \forall R.C_2; C_1 \sqsubseteq \forall R.C_3}; Disj(C_2, C_3); \Rightarrow C_1 \sqsubseteq \forall R.(C_2 \sqcup C_3);$

2.3 Example

The following section describes two definitions from HydrOntology where this antipattern can be found and their English translations. Notice that in each example, the antipattern corresponds to a part of the class definition.

Example Problem about Transitional Water

$Aguas\_de\_Transición \sqsubseteq \forall está\_próxima.Aguas\_Marinas \sqcap \forall está\_próxima.Desembocadura \sqcap = 1 está\_próxima.\top;$

$Transitional\_Water \sqsubseteq \forall is\_nearby.Sea\_Water \sqcap \forall is\_nearby.River\_Mouth \sqcap = 1 is\_nearby.\top;$

Example Solution about Transitional Water

$Aguas\_de\_Transición \sqsubseteq \forall está\_próxima.(Aguas\_Marinas \sqcup Desembocadura) \sqcap = 1 está\_próxima.\top;$

$Transitional\_Water \sqsubseteq \forall is\_nearby.(Sea\_Water \sqcup River\_Mouth) \sqcap = 1 is\_nearby.\top$

Example Problem about Wet Zone

$Zona\_Humeda \sqsubseteq \forall Humedal \sqcap \forall es\_inundada.Aguas\_Marinas \sqcap \forall es\_inundada.Aguas\_Superficiales \sqcap \geq 1 es\_inundada.\top;$

$Wet\_Zone \sqsubseteq \forall Wetlands \sqcap \forall are\_inundated.Sea\_Water \sqcap \forall are\_inundated.Surface\_Water \sqcap \geq 1 are\_inundated.\top;$

Example Solution about Wet Zone

$Zona\_Humeda \sqsubseteq \forall Humedal \sqcap \forall es\_inundada.(Aguas\_Marinas \sqcup Aguas\_Superficiales) \sqcap \geq 1 es\_inundada.\top;$

$Wet\_Zone \sqsubseteq \forall Wetlands \sqcap \forall are\_inundated.(Sea\_Water \sqcup Surface\_Water) \sqcap \geq 1 are\_inundated.\top;$

2.4 Related Resources and Pattern Usage

All the information related to the debugging of the Hydrontology ontology can be found in urlhttp://www.dia.fi.upm.es/ ocorcho/OWLDebugging/. The debugging strategy using this antipattern is described in [2]. Other antipatterns found during the debugging task are defined in [1] and [7]

3 Summary and Future Work

This antipattern can be found in ontologies and may cause inconsistency problems. We provide a solution to it, so that it can be used by domain experts to debug their ontologies. In the future, we aim at implementing additional tools to help in the identification of antipatterns in well-known inconsistent ontologies (e.g., TAMBIS). For the time being we have started applying the OPPL language [4] for this task, with promising results.

References

1. Corcho O., Roussey C., Vilches Blazquez L.M.: Catalogue of Anti-Patterns for formal Ontology debugging. In Proceedings of Construction d'ontologies : vers un guide des bonnes pratiques, AFIA 2009, Hammamet, Tunisie. (2009).
2. Corcho O., Roussey C., Vilches Blazquez L.M.: Pattern-based OWL Ontology Debugging Guidelines. In Proceedings of 1st Workshop on Ontology Patterns (WOP2009), Washington DC, USA. (2009).
3. Horridge M, Parsia B, Sattler U.: Laconic and Precise Justifications in OWL. In Proceedings of the 7th International Semantic Web Conference (ISWC), Karlsruhe, Germany; LNCS 5318: 323-338. (2008).
4. Iannone L, Rector A, Stevens R.: Embedding Knowledge Patterns into OWL. In proceedings of the 6th European Semantic Web Conference (ESWC2009), Crete, Greece. The Semantic Web: Research and Applications (2009), pp. 218-232
5. Kalyanpur A, Parsia B, Sirin E, Cuenca-Grau B.: Repairing Unsatisfiable Classes in OWL Ontologies. In Proceedings of the 3rd European Semantic Web Conference (ESWC), Budva, Montenegro; LNCS 4011: 170-184 (2006)
6. Koenig A.: Patterns and Antipatterns. Journal of Object-Oriented Programming 8(1):46-48. (1995)
7. Roussey C., Corcho O., Vilches Blazquez L.M.: A Catalogue of OWL Ontology AntiPatterns. In Proceedings of the Fifth International Conference on Knowledge Capture KCAP 2009, Yolanda Gil, Natasha Noy ed. Redondo Beach, California, USA. ISBN 978-1-60558-658-8. pp. 205-206 (2009)
8. Stuckenschmidt H.: Debugging OWL Ontologies - a Reality Check. In Proceedings of the 6th International Workshop on Evaluation of Ontology-based Tools and the Semantic Web Service Challenge (EON-SWSC-2008), Tenerife, Spain. (2008).
9. Vilches-Blázquez LM, Bernabé-Poveda MA, Suárez-Figueroa MC, Gómez-Pérez A, Rodrguez-Pascual AF: Towntology & hydrOntology: Relationship between Urban and Hydrographic Features in the Geographic Information Domain. In Ontologies for Urban Development. Studies in Computational Intelligence, vol. 61, Springer: 73-84. (2007)