EMPIRICAL EVALUATION OF MLT-OWL

Technical Report, v1.0, June 2016.

Freddy Brasileiro[a], João Paulo A. Almeida[a], Victorio A. Carvalho[a,b] and Giancarlo Guizzardi[a]

[a]Ontology & Conceptual Modeling Research Group (NEMO), Federal University of Espírito Santo (UFES), Vitória, ES, Brazil.

[b]Research Group in Applied Informatics, Informatics Department, Federal Institute of Espírito Santo (IFES), Colatina, ES, Brazil.

freddybrasileiro@gmail.com, jpalmeida@ieee.org, victorio@ifes.edu.br, gguizzardi@inf.ufes.br

Empirical Evaluation of MLT-OWL. Technical Report (version 1.0).

ABSTRACT

This document aims to show how the MLT-OWL approach proposed in [1] is capable of preventing a number of issues found in Wikidata multi-level taxonomical hierarchies. As reported in [2], there are over 22,000 occurrences of three anti-patterns related to multi-level modeling in current Wikidata content; all these occurrences would have been prevented by using MLT-OWL. We show how each of three anti-patterns can be detected using MLT-OWL. In addition, we demonstrate how MLT-OWL approach could be used to complete valid models using fragments of Wikidata content.

CONTENTS

LIST OF FIGURES

1. INTRODUCTION

This document aims to show how the MLT-OWL approach proposed in [1] is capable of preventing a number of issues found in Wikidata multi-level taxonomical hierarchies. As reported in [2], there are over 22,000 occurrences of three anti-patterns related to multi-level modeling in current Wikidata content; all these occurrences would have been prevented by using MLT-OWL. We show how each of three anti-patterns can be detected using MLT-OWL. In addition, we demonstrate how MLT-OWL approach could be used to complete valid models using fragments of Wikidata content.

The full set of integrity constraints and derivation rules described partially in [1] are implemented in a tool available at https://github.com/freddybrasileiro/mlt-owl. The tool receives as input an OWL file describing a domain ontology. It detects violations of MLT rules and, in the case of valid models, it produces an OWL output file containing the original domain ontology plus derived information following MLT rules [3]. At least one the domain entities in the input domain ontology should be declared to be an instance of or a subclass of the basic MLT classes (the leaf classes in Fig. 3 of [1].)

Each of the presented examples is serialized in a separate OWL file available at https://github.com/freddybrasileiro/mlt-owl/tree/master/br.ufes.inf.nemo.mlt.web/resources/examples.

This document is further structured as follows: Section 2 presents the three anti-patterns discussed in [2], showing example ontologies extracted from Wikidata content in which the anti-patterns occur; the ontologies are analyzed using the MLT-OWL tool, which detects the multi-level modeling problems. Section 3 presents other examples in which the tool is used to derive additional information for the domain ontology based on the MLT rules.

2. AVOIDING ANTI-PATTERNS

A. ANTI-PATTERN 1 (AP1)

Figure 1 illustrates the Anti-Pattern 1 (AP1) that looks for pairs of items (A, Z) such that the second one (Z) is simultaneously a subclass of A and an instance of A. This anti-pattern can appear under many configurations, i.e., subclass (Z) can be a direct subclass of A or there may be a chain of subclass of properties between the involved items. The fragment illustrated in Figure 1 (concerning Tim Berners-Lee’s professional occupation) includes two occurrences of this anti-pattern with chains of subclass of properties of length 2 and 3. Regardless of the size of this chain, the occurrence of this pattern prevents stratification into classification levels, and creates a formal contradiction: classes A and Z would have be simultaneously at the same level (because they are related by specialization) and at adjacent levels (because they are related by instantiation).

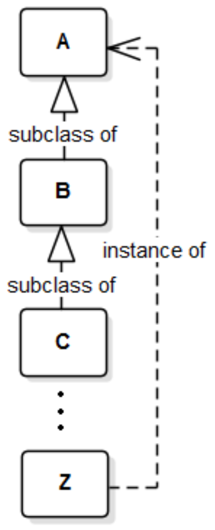


Figure 1. Illustration of Anti-Pattern 1

Following we present three scenarios found in Wikidata for AP1.

I. TIM BERNERS-LEE'S PROFESSION

In Figure 2 we present the example about *Tim Berners-Lee* profession. This example is serialized in the file named www-fig3.owl.

In addition to the information presented in the example, we added the information that Tim Berners-Lee is an instance of *mlt:TokenIndividual*, since we know that he is a concrete individual and he has no instances.

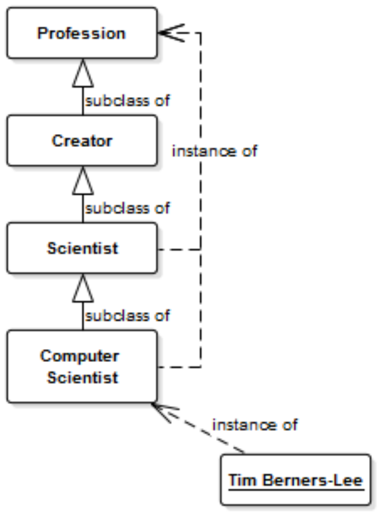
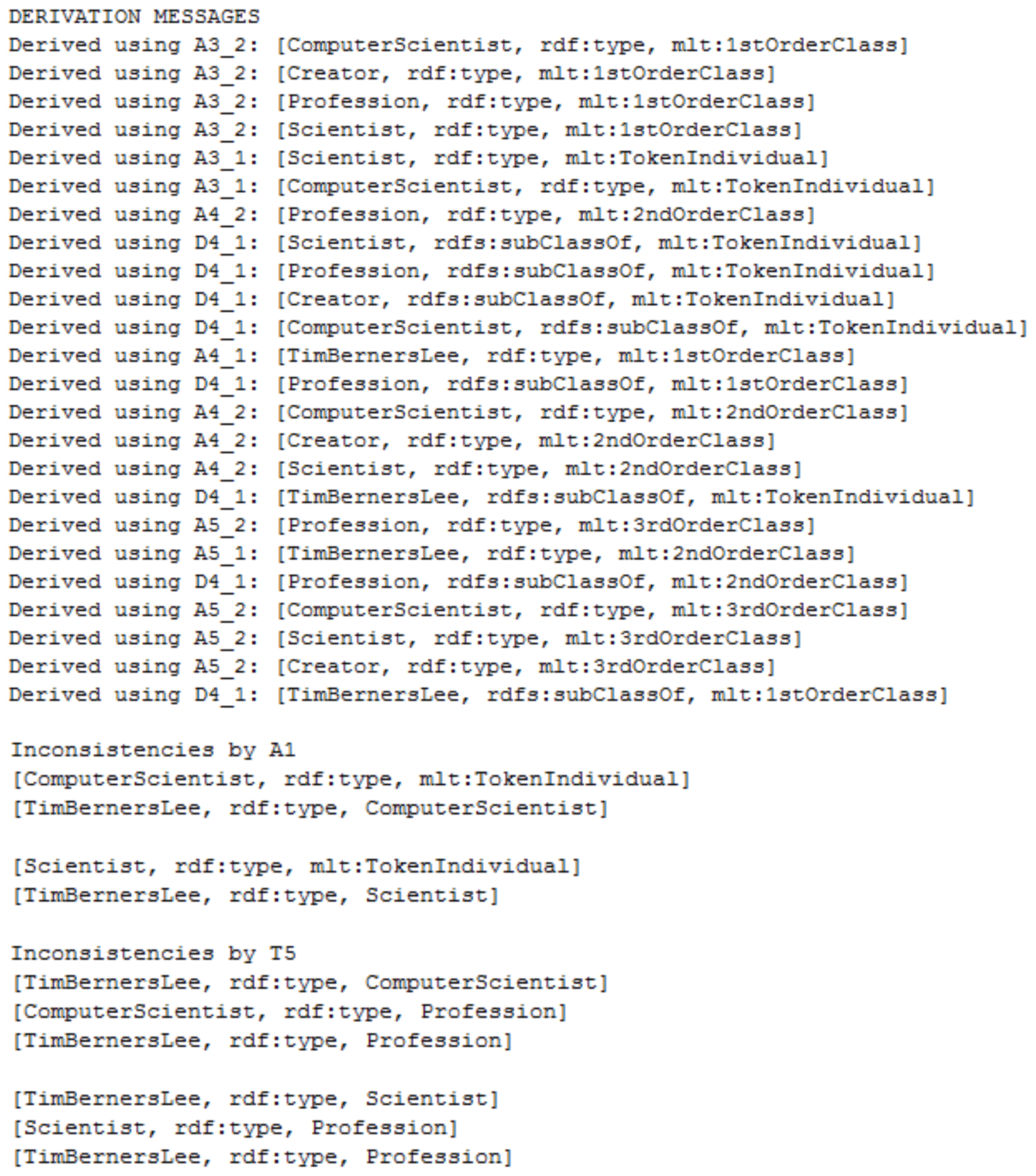


Figure 2. Wikidata information about Tim Berners-Lee and his professional occupation

Selecting the file containing this example, our implementation shows the messages presented in Figure 3.



```
DERIVATION MESSAGES
Derived using A3_2: [ComputerScientist, rdf:type, mlt:1stOrderClass]
Derived using A3_2: [Creator, rdf:type, mlt:1stOrderClass]
Derived using A3_2: [Profession, rdf:type, mlt:1stOrderClass]
Derived using A3_2: [Scientist, rdf:type, mlt:1stOrderClass]
Derived using A3_1: [Scientist, rdf:type, mlt:TokenIndividual]
Derived using A3_1: [ComputerScientist, rdf:type, mlt:TokenIndividual]
Derived using A4_2: [Profession, rdf:type, mlt:2ndOrderClass]
Derived using D4_1: [Scientist, rdfs:subClassOf, mlt:TokenIndividual]
Derived using D4_1: [Profession, rdfs:subClassOf, mlt:TokenIndividual]
Derived using D4_1: [Creator, rdfs:subClassOf, mlt:TokenIndividual]
Derived using D4_1: [ComputerScientist, rdfs:subClassOf, mlt:TokenIndividual]
Derived using A4_1: [TimBernersLee, rdf:type, mlt:1stOrderClass]
Derived using D4_1: [Profession, rdfs:subClassOf, mlt:1stOrderClass]
Derived using A4_2: [ComputerScientist, rdf:type, mlt:2ndOrderClass]
Derived using A4_2: [Creator, rdf:type, mlt:2ndOrderClass]
Derived using A4_2: [Scientist, rdf:type, mlt:2ndOrderClass]
Derived using D4_1: [TimBernersLee, rdfs:subClassOf, mlt:TokenIndividual]
Derived using A5_2: [Profession, rdf:type, mlt:3rdOrderClass]
Derived using A5_1: [TimBernersLee, rdf:type, mlt:2ndOrderClass]
Derived using D4_1: [Profession, rdfs:subClassOf, mlt:2ndOrderClass]
Derived using A5_2: [ComputerScientist, rdf:type, mlt:3rdOrderClass]
Derived using A5_2: [Scientist, rdf:type, mlt:3rdOrderClass]
Derived using A5_2: [Creator, rdf:type, mlt:3rdOrderClass]
Derived using D4_1: [TimBernersLee, rdfs:subClassOf, mlt:1stOrderClass]

Inconsistencies by A1
[ComputerScientist, rdf:type, mlt:TokenIndividual]
[TimBernersLee, rdf:type, ComputerScientist]

[Scientist, rdf:type, mlt:TokenIndividual]
[TimBernersLee, rdf:type, Scientist]

Inconsistencies by T5
[TimBernersLee, rdf:type, ComputerScientist]
[ComputerScientist, rdf:type, Profession]
[TimBernersLee, rdf:type, Profession]

[TimBernersLee, rdf:type, Scientist]
[Scientist, rdf:type, Profession]
[TimBernersLee, rdf:type, Profession]
```

Figure 3. Output for the example of Tim Berners-Lee profession from Wikidata

Note that two occurrences of inconsistency (detected with SPARQL). In the first one, we have from A2 that *ComputerScientist, Scientist, Creator* and *Profession* are derived as instances of *mlt:1stOrderClass*, since they have a *mlt:TokenIndividual* as its instance (*Tim Berners-Lee*). However, also from A2 we have that *ComputerScientist* is an instance of *mlt:TokenIndividual*, since it is an instance of *Profession*. This scenario is not allowed according to A1, where instances of *mlt:TokenIndividual* cannot have instances. In the second occurrence, the message shows that this model violates T5, about strict metamodeling.

Now, using Jena to check consistency of the model according to MLT rules expressed through OWL axioms, we have that this model is also found inconsistent (Figure 4). This is due to the fact that the basic types of MLT are disjoint (T4). Thus, since *ComputerScientist* is derived both as instance of *mlt:TokenIndividual* and *mlt:1stOrderClass*, this model is inconsistent according to MLT.

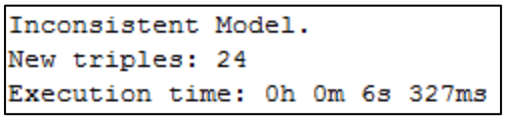

```
Inconsistent Model.
New triples: 24
Execution time: 0h 0m 6s 327ms
```

Figure 4. Final message for the example of Tim Berners-Lee profession from Wikidata

II. EARTHQUAKES

In Figure 5 we present the example about *earthquakes*. This example is serialized in the file named www-fig5.owl.

In addition to the information presented in the example, we added the information that *earthquake* is an instance of *mlt:1stOrderClass*.

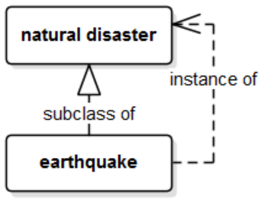


Figure 5. Scenario about earthquakes found in Wikidata for AP1

Selecting the file containing this example, our implementation shows the messages presented in Figure 6.

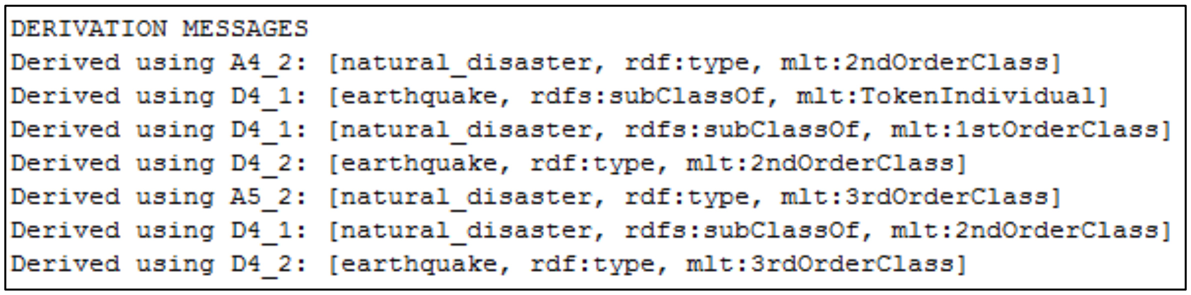
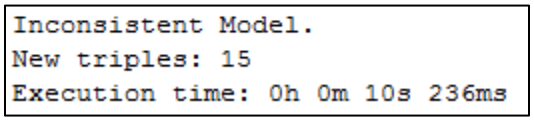

```
DERIVATION MESSAGES
Derived using A4_2: [natural_disaster, rdf:type, mlt:2ndOrderClass]
Derived using D4_1: [earthquake, rdfs:subClassOf, mlt:TokenIndividual]
Derived using D4_1: [natural_disaster, rdfs:subClassOf, mlt:1stOrderClass]
Derived using D4_2: [earthquake, rdf:type, mlt:2ndOrderClass]
Derived using A5_2: [natural_disaster, rdf:type, mlt:3rdOrderClass]
Derived using D4_1: [natural_disaster, rdfs:subClassOf, mlt:2ndOrderClass]
Derived using D4_2: [earthquake, rdf:type, mlt:3rdOrderClass]
```

Figure 6. Derivation messages for the example of earthquakes from Wikidata

Note that there are no messages of inconsistency. However, using Jena to check consistency of the model according to MLT rules expressed through OWL axioms, we have that this model is inconsistent (as shown in Figure 7). This is due to the fact that the basic types of MLT are disjoint

(T4). Thus, since *earthquake* is derived both as instance of *mlt:TokenIndividual* and *mlt:2ndOrderClass*, this model is inconsistent according to MLT.



```
Inconsistent Model.
New triples: 15
Execution time: 0h 0m 10s 236ms
```

Figure 7. Final message for the example of earthquakes from Wikidata

B. ANTI-PATTERN 2 (AP2)

A second anti-pattern (AP2) is illustrated in Figure 8. In this case, we have that an item (*C*) has two direct super classes (*A* and *B*) such that one of them is an instance of the other (*B is instance of A*). Similarly to AP1, the occurrences of AP2 present logical inconsistencies that arise from the violation of the strict metamodeling principle. In this case, all *instances of C* are also *instances of A* and *B*. However, *instances of B* cannot be *instances of A* since *B* is itself *instance of A*.

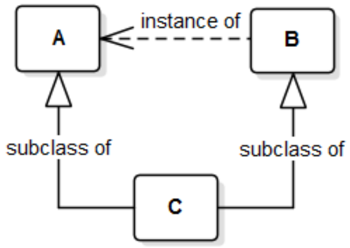


Figure 8. Illustration of Anti-Pattern 2

I. EXCAVATOR

In Figure 9 we present the example about *excavator*. This example is serialized in the file www-fig8.owl.

In addition to the information presented in the example, we added the information *crawler excavator* is an instance of *mlt:1stOrderClass*.

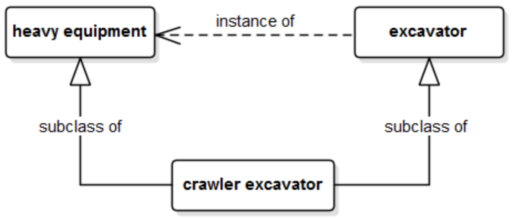
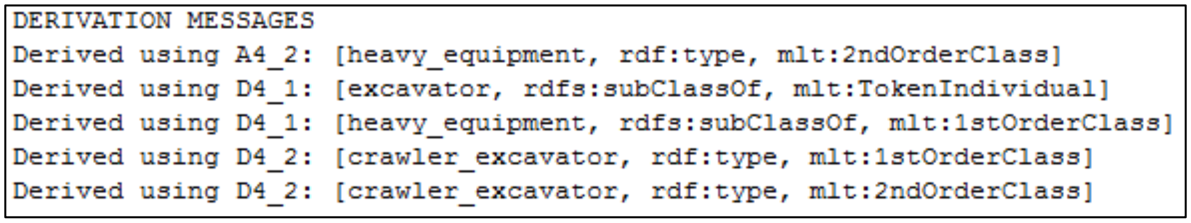


Figure 9. Scenario about excavators found in Wikidata for AP2

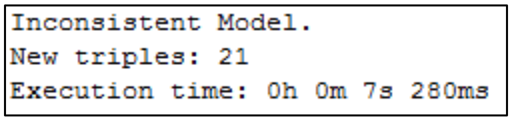
Selecting the file containing this example, our implementation shows the messages presented in Figure 10.



```
DERIVATION MESSAGES
Derived using A4_2: [heavy_equipment, rdf:type, mlt:2ndOrderClass]
Derived using D4_1: [excavator, rdfs:subClassOf, mlt:TokenIndividual]
Derived using D4_1: [heavy_equipment, rdfs:subClassOf, mlt:1stOrderClass]
Derived using D4_2: [crawler_excavator, rdf:type, mlt:1stOrderClass]
Derived using D4_2: [crawler_excavator, rdf:type, mlt:2ndOrderClass]
```

Figure 10. Derivation messages for the example of excavators from Wikidata

Note that there are no messages of inconsistency. However, using Jena to check consistency of the model according to MLT rules expressed through OWL axioms, we have that this model is inconsistent (as shown in Figure 11). This is due to the fact that the basic types of MLT are disjoint (T4). Thus, since *heavy equipment* is derived both as instance of *mlt:1stOrderClass* and *mlt:2ndOrderClass*, this model is inconsistent according to MLT.



```
Inconsistent Model.
New triples: 21
Execution time: 0h 0m 7s 280ms
```

Figure 11. Final message for the example of excavators from Wikidata

C. ANTI-PATTERN 3 (AP3)

Finally, a third anti-pattern (AP3) is illustrated in Figure 12. This anti-pattern represents cases in which the anti-transitivity of the instance of relation is violated, making stratification unfeasible. In the case depicted in Figure 12, *C* would have to be simultaneously one and two classification levels below *A*.

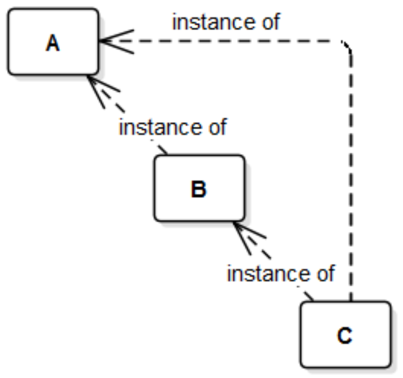


Figure 12. Illustration of Anti-Pattern 3

I. URBAN PARK

In Figure 13 we present the example about the *Central Park*. This example is serialized in the file named www-fig10.owl.

In addition to the information presented in the example, we added the information that *Central Park* is an instance of *mlt:TokenIndividual*.

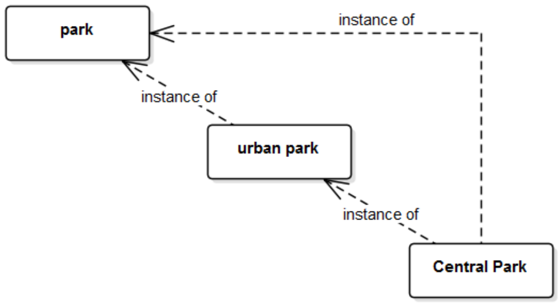
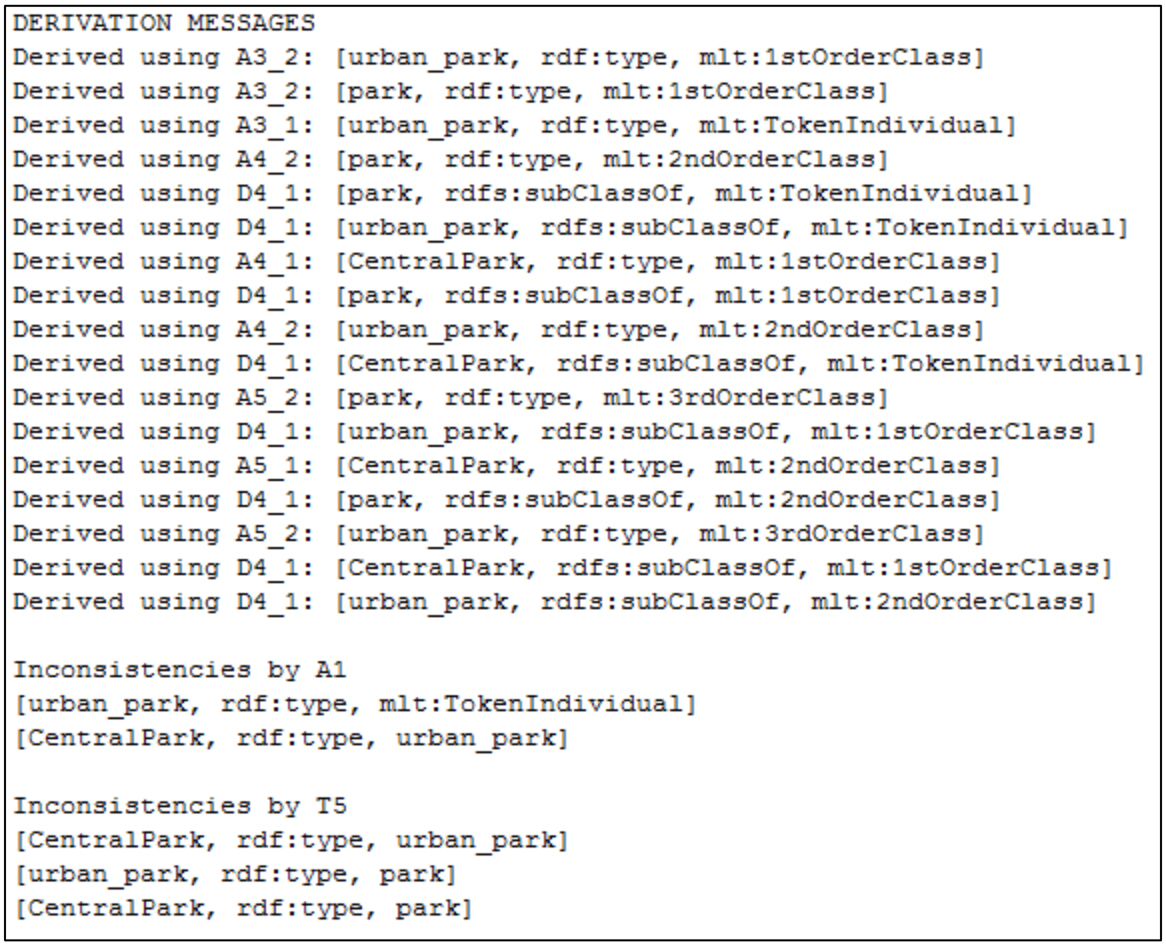


Figure 13. Scenario about urban parks found in Wikidata for AP3

Selecting the file containing this example, our implementation shows the messages presented in Figure 14.

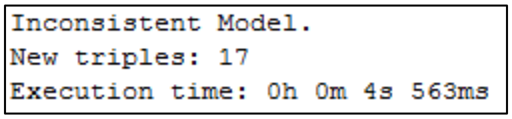


```
DERIVATION MESSAGES
Derived using A3_2: [urban_park, rdf:type, mlt:1stOrderClass]
Derived using A3_2: [park, rdf:type, mlt:1stOrderClass]
Derived using A3_1: [urban_park, rdf:type, mlt:TokenIndividual]
Derived using A4_2: [park, rdf:type, mlt:2ndOrderClass]
Derived using D4_1: [park, rdfs:subClassOf, mlt:TokenIndividual]
Derived using D4_1: [urban_park, rdfs:subClassOf, mlt:TokenIndividual]
Derived using A4_1: [CentralPark, rdf:type, mlt:1stOrderClass]
Derived using D4_1: [park, rdfs:subClassOf, mlt:1stOrderClass]
Derived using A4_2: [urban_park, rdf:type, mlt:2ndOrderClass]
Derived using D4_1: [CentralPark, rdfs:subClassOf, mlt:TokenIndividual]
Derived using A5_2: [park, rdf:type, mlt:3rdOrderClass]
Derived using D4_1: [urban_park, rdfs:subClassOf, mlt:1stOrderClass]
Derived using A5_1: [CentralPark, rdf:type, mlt:2ndOrderClass]
Derived using D4_1: [park, rdfs:subClassOf, mlt:2ndOrderClass]
Derived using A5_2: [urban_park, rdf:type, mlt:3rdOrderClass]
Derived using D4_1: [CentralPark, rdfs:subClassOf, mlt:1stOrderClass]
Derived using D4_1: [urban_park, rdfs:subClassOf, mlt:2ndOrderClass]

Inconsistencies by A1
[urban_park, rdf:type, mlt:TokenIndividual]
[CentralPark, rdf:type, urban_park]

Inconsistencies by T5
[CentralPark, rdf:type, urban_park]
[urban_park, rdf:type, park]
[CentralPark, rdf:type, park]
```

Figure 14. Derivation messages for the example of urban parks from Wikidata

Note that two occurrences of inconsistency are shown. We have from A3 that *urban park* and *park* are derived as instances of *mlt:1stOrderClass*, since they have a *mlt:TokenIndividual* as its instance (*Central Park*). However, also from A3 we have that *urban park* is an instance of *mlt:TokenIndividual*, since it is an instance of *park*. This scenario is not allowed according to A1, where instances of *mlt:TokenIndividual* cannot have instances. In the second occurrence, the message shows that this model violates T5, about strict metamodeling.

Now, using Jena to check consistency of the model according to MLT rules expressed through OWL axioms, we have that this model is inconsistent (as shown in Figure 15). This is due to the fact that the basic types of MLT are disjoint (T4). Thus, since *urban park* is derived both as instance of *mlt:TokenIndividual* and *mlt:1stOrderClass*, this model is inconsistent according to MLT.



```
Inconsistent Model.
New triples: 17
Execution time: 0h 0m 4s 563ms
```

Figure 15. Final message for the example of urban parks from Wikidata

3. APPLYING DERIVATION RULES FOR VALID MODELS

A. BIOLOGY TAXONOMY

In Figure 16, we present the example about the lion *Cecil*. This example is presented in [1] and it is similar to the one presented in [2]. This example is serialized in the file named iswc-fig5.owl.

In addition to the information presented in the example, we added the information that *Cecil* is an instance of *mlt:TokenIndividual*, since we know that he is a concrete individual and he has no instances.

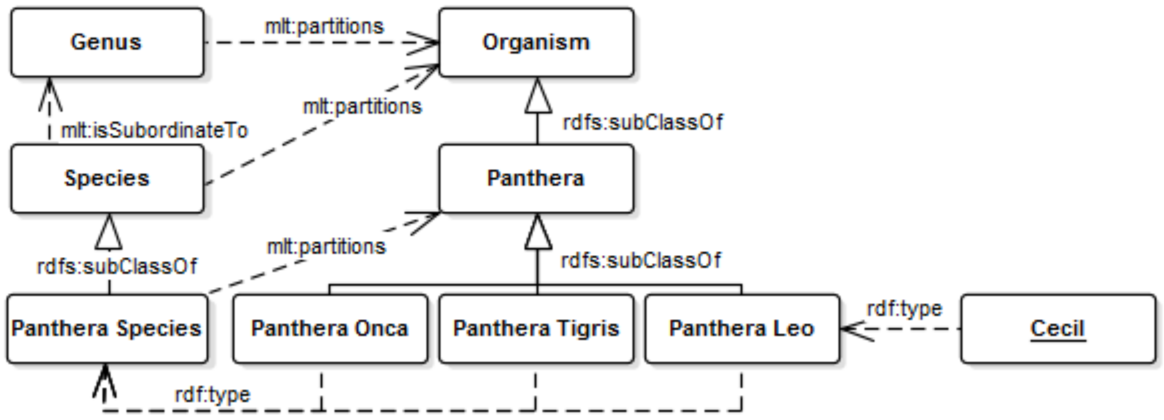
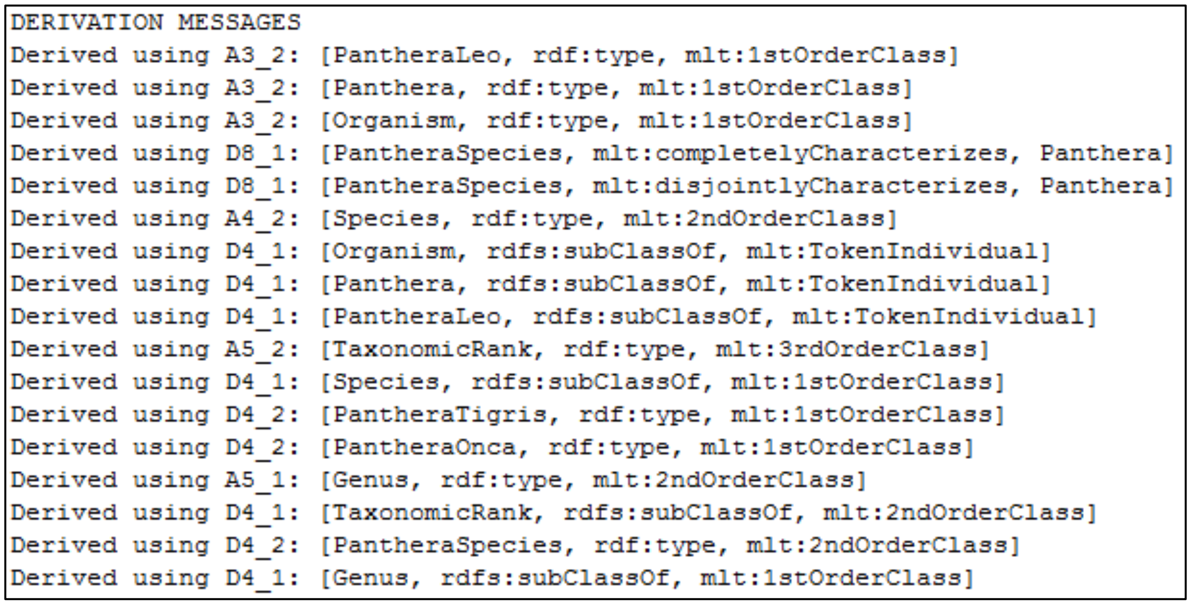


Figure 16. Short representation for Taxonomic Biological Domain with MLT relations

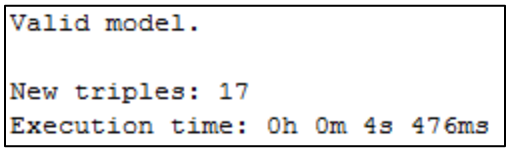
Selecting the file containing this example, our implementation shows the messages presented in Figure 17.



```
DERIVATION MESSAGES
Derived using A3_2: [PantheraLeo, rdf:type, mlt:1stOrderClass]
Derived using A3_2: [Panthera, rdf:type, mlt:1stOrderClass]
Derived using A3_2: [Organism, rdf:type, mlt:1stOrderClass]
Derived using D8_1: [PantheraSpecies, mlt:completelyCharacterizes, Panthera]
Derived using D8_1: [PantheraSpecies, mlt:disjointlyCharacterizes, Panthera]
Derived using A4_2: [Species, rdf:type, mlt:2ndOrderClass]
Derived using D4_1: [Organism, rdfs:subClassOf, mlt:TokenIndividual]
Derived using D4_1: [Panthera, rdfs:subClassOf, mlt:TokenIndividual]
Derived using D4_1: [PantheraLeo, rdfs:subClassOf, mlt:TokenIndividual]
Derived using A5_2: [TaxonomicRank, rdf:type, mlt:3rdOrderClass]
Derived using D4_1: [Species, rdfs:subClassOf, mlt:1stOrderClass]
Derived using D4_2: [PantheraTigris, rdf:type, mlt:1stOrderClass]
Derived using D4_2: [PantheraOnca, rdf:type, mlt:1stOrderClass]
Derived using A5_1: [Genus, rdf:type, mlt:2ndOrderClass]
Derived using D4_1: [TaxonomicRank, rdfs:subClassOf, mlt:2ndOrderClass]
Derived using D4_2: [PantheraSpecies, rdf:type, mlt:2ndOrderClass]
Derived using D4_1: [Genus, rdfs:subClassOf, mlt:1stOrderClass]
```

Figure 17. Derivation messages for the example of Biological Taxonomy with MLT relations

Note that, as said in [2], this is an example of a valid model. From the information that *Cecil* is an instance of *mlt:TokenIndividual*, all other entities are derived as subclasses or instances of MLT basic types. Figure 18 shows the final message informing that the model is valid, the number of new triples and the execution time.



```
Valid model.

New triples: 17
Execution time: 0h 0m 4s 476ms
```

Figure 18. Final message for the example of Taxonomic Biological Domain with MLT relations

B. COMPANIES EMPLOYEE

In Figure 19, we present the example about employee roles in a company. This example is presented in [1] and is serialized in the file named iswc-fig6.owl.

In addition to the information presented in the example, we added the information that *Employee* is an instance of *mlt:1stOrderClass*, since we know that he is a concrete individual and he has no instances.

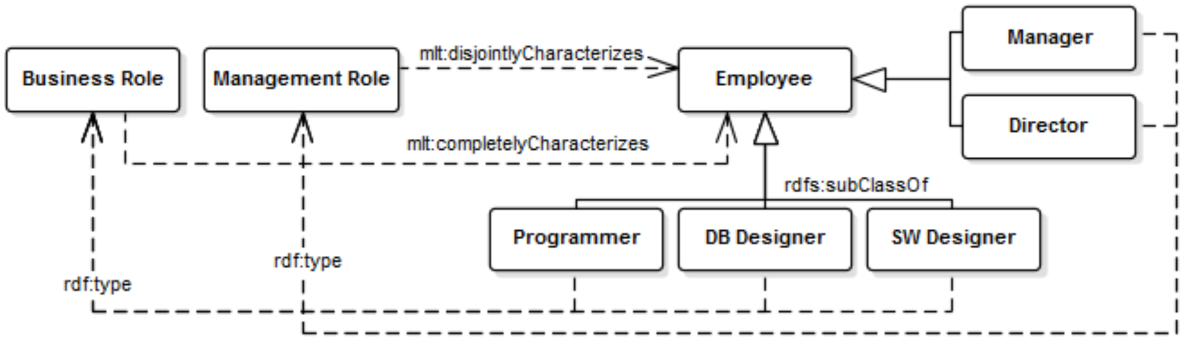
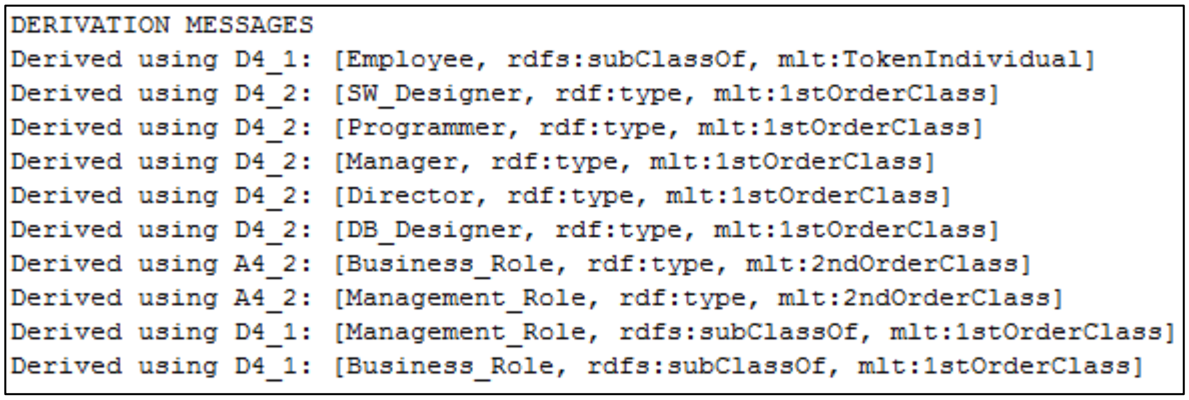


Figure 19. Short representation for Company Employee domain with MLT relations

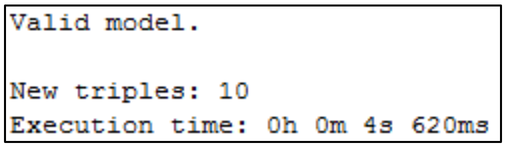
Selecting the file containing this example, our implementation shows the messages presented in Figure 20.



```
DERIVATION MESSAGES
Derived using D4_1: [Employee, rdfs:subClassOf, mlt:TokenIndividual]
Derived using D4_2: [SW_Designer, rdf:type, mlt:1stOrderClass]
Derived using D4_2: [Programmer, rdf:type, mlt:1stOrderClass]
Derived using D4_2: [Manager, rdf:type, mlt:1stOrderClass]
Derived using D4_2: [Director, rdf:type, mlt:1stOrderClass]
Derived using D4_2: [DB_Designer, rdf:type, mlt:1stOrderClass]
Derived using A4_2: [Business_Role, rdf:type, mlt:2ndOrderClass]
Derived using A4_2: [Management_Role, rdf:type, mlt:2ndOrderClass]
Derived using D4_1: [Management_Role, rdfs:subClassOf, mlt:1stOrderClass]
Derived using D4_1: [Business_Role, rdfs:subClassOf, mlt:1stOrderClass]
```

Figure 20. Derivation messages for the example of Companies Employee with MLT relations

Note that this is an example of a valid model. From the information that *Employee* is an instance of *mlt:1stOrderClass*, all other entities are derived as subclasses or instances of MLT basic types. Figure 21 shows the final message informing that the model is valid, the number of new triples and the execution time.



```
Valid model.

New triples: 10
Execution time: 0h 0m 4s 620ms
```

Figure 21. Final message for the example of Companies Employee with MLT relations

4. REFERENCES

[1] F. Brasileiro, J. P. A. Almeida, V. A. Carvalho, and G. Guizzardi, “Expressive Multi-Level Modeling for the Semantic Web [submitted].”

[2] F. Brasileiro, J. P. A. Almeida, V. A. Carvalho, and G. Guizzardi, “Applying a Multi-Level Modeling Theory to Assess Taxonomic Hierarchies in Wikidata,” in *Wiki Workshop 2016 at 25th Int. Conference Companion on World Wide Web*, 2016, pp. 975–980.

[3] V. A. Carvalho and J. P. A. Almeida, “Towards a Well-Founded Theory for Multi-Level Conceptual Modelling [submitted],” 2016.