OntoClean in OWL with a DL reasoner – A tutorial

Zola Mahlaza and C. Maria Keet

(Based on an OE mini-project by Todii Mashoko, Siseko Neti, and Banele Matsebula)

The aim of this tutorial is to illustrate how the OntoClean methodology can be used in Protégé with OWL and its reasoner, based on the OntOWLClean approach proposed in [Welty, 2006]. In particular, it focuses on the following tasks:

- Punning[1] an ontology in preparation for OntoClean.
- Assigning meta-properties to classes in OWL.
- Discovering inconsistencies in a taxonomy, and hints for fixing the hierarchy.

The rest of the document is structured such that Section 1 presents an overview of OntoClean and Section 2 presents our example ontology, the manner in which OntoClean can be followed within Protégé, and a limited number of errors that should be discovered from the provided ontology.

1 OntoClean

OntoClean is a philosophy-based methodology for validating the correctness and consistency of an ontology's taxonomy. It is based on general notions that are drawn from philosophy, which are Rigidity, Identity, Unity, and Dependence. The methodology is made up of two phases. The first phase involves annotating all the classes within an ontology with labels of the meta-properties referring to the four philosophical notions. The second phase deals with the checking of subsumption relationships of the ontology based on the predefined OntoClean constraints, which in this document are also referred to as rules.

In the remainder of this section, we provide a brief recap to the four philosophical concepts, the OntoClean meta-property annotation symbols, and the constraints for each of them. A larger summary (1.5 pages) is included as section 5.2.2 of the textbook [Keet, 2018], a full overview of OntoClean is described in the handbook [Guarino and Welty, 2009], which build upon foundations presented in [Guarino and Welty, 2000a, Guarino and Welty, 2000b].

[1]`https://www.w3.org/2007/OWL/wiki/Punning`

Rigidity refers to an entity's property that are essential to that entity (i.e. it must be true of it in every possible situation). For instance, the property of having walls is essential to a house. Every house must have walls in every possible situation. In the event that they are demolished then you no longer have a house. Identity refers to the capability to identify individual entities in the world as being the same or different. Unity refers to the ability to describe the parts and boundaries of objects, and thus to know which parts constitute an object, which parts do not, and under what circumstances is the object a whole. Dependence is the relationship between entities whereby one will exist solely on the existence of the other.

All the entities within an ontology must be assigned with meta-properties and labelled with the letter denoting the meta-property; more precisely: (**I**) for Identity, (**U**) for unity, (**D**) for dependence and (**R**) for Rigidity. Each of these labels preceded with a $+$, $-$, or $\sim$ symbol, where $(+)$ means the entity is what the letter denotes, $(-)$ means the entity is not what the letter denotes, and $(\sim)$ means 'anti' (may or may not be) to what the letter denotes.

The assignment of meta-properties is useful because there are constraints that the taxonomy must not violate. For instance, when we have two properties `x` and `z`, where `z` subsumes `x` then we know that if `z` is anti-rigid ($\sim$R) then `x` must be anti-rigid (Rigidity constraint), if `z` carries an identity criterion (+I) then `x` must carry the same criterion (Identity constraint[2]), if `z` carries a unity criterion (+U) then `x` must carry the same criterion and if `z` has anti-unity, then `x` must also have anti-unity (Unity constraints), and if `z` is dependent (+D) on a certain property `y` then `x` is dependent on property `y` (Dependence constraint).

2 OntoClean in Protégé

In this section we will specify where to download the required software, describe the ontology we will use to illustrate OntoClean, introduce how to pun the ontology in Protégé in preparation for OntoClean, and present an exercise on assigning meta-properties in the tutorial ontology.

2.1 What to download

You will need the following material for this tutorial:

- Protégé 5.x, which can be downloaded from `https://protege.stanford.edu/`.
- OntoClean and AmountOfMatter tutorial ontologies:
  - `ontoclean-dl.owl` (OntoClean ontology)[3]

[2]This is not true in the case of the "own" identity criteria

[3]The OWL-DL ontology developed by [Welty, 2006] is no longer available through the OntoClean website at `http://www.ontoclean.org/`, therefore we provide a cached version.

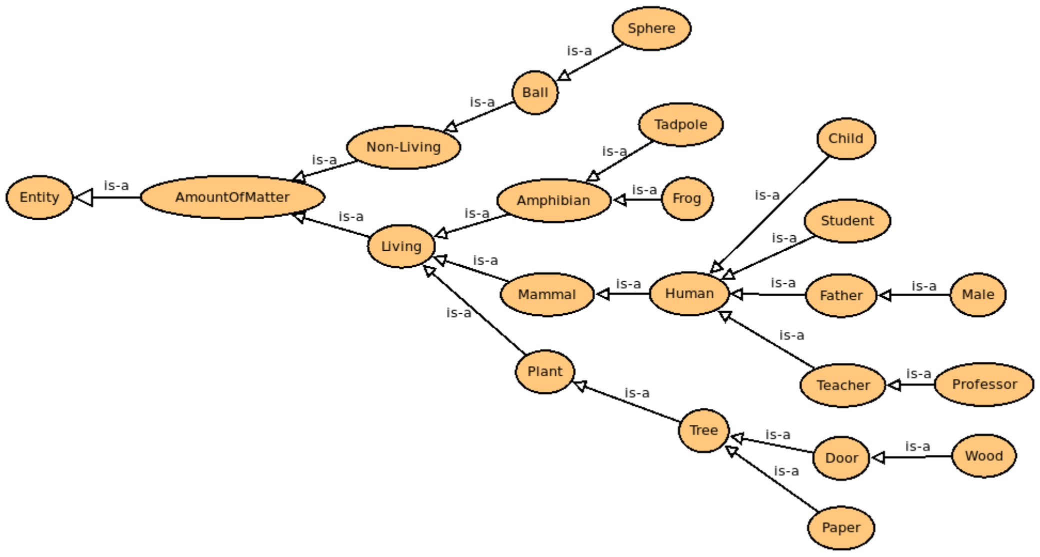


Figure 1: Tutorial ontology with ontological inconsistencies.

- `OntocleanTutorialOntology.owl` (AmountOfMatter ontology, which is the domain ontology that is to be 'cleaned up')
- `OntocleanTutorialOntologyPunned.owl` (Punned AmountOfMatter ontology)
- `OntocleanTutorialOntologyPunnedMetaProperties.owl` (Punned AmountOfMatter ontology with assigned meta-properties)

which can be downloaded from the textbook's resources page at `https://people.cs.uct.ac.za/~mkeet/OEbook/ontologies/`

Please note that another version of Protégé can be used, however, the interface may use different terms and have a different layout and look-and-feel.

2.2 AmountOfMatter taxonomy

The AmountOfMatter tutorial ontology (see file `OntocleanTutorialOntology.owl`) that will be used to illustrate how to follow OntoClean is shown in Figure 1. The tutorial ontology has deliberate taxonomic issues that need to be resolved using OntoClean. In the ontology, *AmountOfMatter* can be considered as any entity and may be a *Non-Living* or *Living* object. An example of *Non-Living* object, according to the ontology, is a *Ball*. Familiarise yourself with the ontology in Protégé by doing Task 1.

2.3 Punning the tutorial ontology

The first task is to push the ontology's Tbox into the ABox [Welty, 2006]. In particular, you must create an Individual with the same name as the class (bears the same

Task 1

Open the tutorial ontology `OntocleanTutorialOntology.owl` in Protégé, import the OntoClean OWL-DL ontology `ontoclean-dl.owl`, and explore.

IRI), for each of the classes on the tutorial ontology. These Individuals must be of type `ontoclean:Class`. An example is shown in Figure 2 where there are 21 Individuals of type `ontoclean:Class`. For each individual, you must then specify its "subclasses" through `ontoclean:hasSubClass` relation. For instance, *AmountOfMatter* has the subclasses *Non-Living* and *Living* as shown in Figure 2. Familiarise yourself with how to do this by conduction Task 2. You can verify whether you've done this task correctly by cross-checking with the provided `OntocleanTutorialOntologyPunned.owl` file.

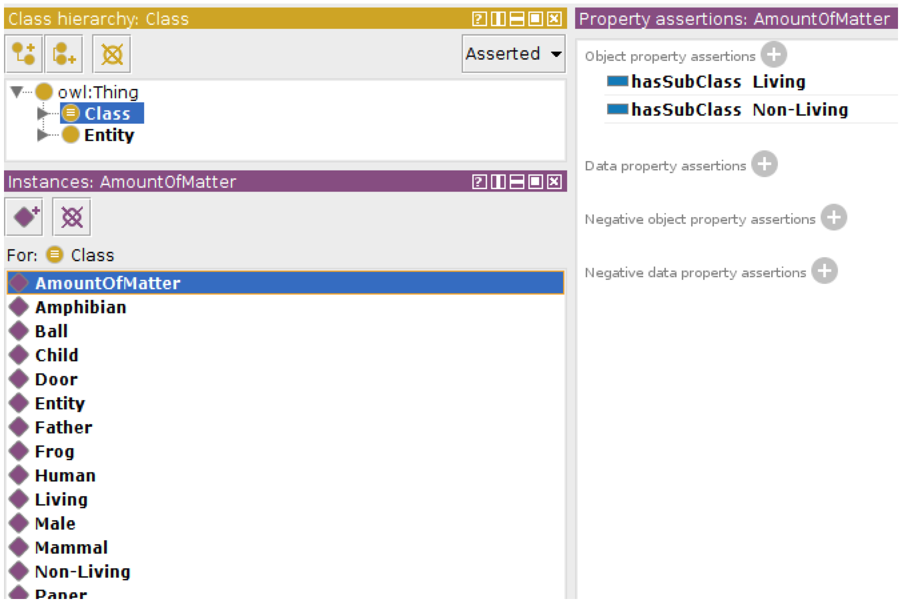
Task 2

1. Continue from Task 1.

2. For each class in the ontology, create its corresponding Individual with the same name as the class.

3. Make each newly created individual an instance of `ontoclean:Class`.

4. Specify the "subclasses" for each Individual, using the `ontoclean:hasSubClass` object property.


Class hierarchy: Class
Asserted
owl:Thing
Class
Entity
Instances: AmountOfMatter
For: Class
Property assertions: AmountOfMatter
Object property assertions
Data property assertions
Negative object property assertions
Negative data property assertions

AmountOfMatter
Amphibian
Ball
Child
Door
Entity
Father
Frog
Human
Living
Male
Mammal
Non-Living
Paper

hasSubClass Living
hasSubClass Non-Living

Figure 2: Individuals of type `ontoclean:Class` and *AmountOfMatter*'s two subclasses asserted in the ABox with `ontoclean:hasSubClass`.

2.4 Assigning the meta-properties

Once the ontology has been punned, you can assign meta-properties to each individual. This requires you to first decide on the meta-properties for each class/OWLfile-individual. Once that is decided, you can use Protégé to assign meta-properties to each individual by setting its type to the appropriate subclasses of `ontoclean:Class`. For instance, you could deem that *Paper* is dependent (+D) by assigning it the *Dependent* meta-property, which practically amounts to adding the assertion that *Paper* is an instance of `ontoclean:DependentClass` as shown in Figure 3. Carry out Task 3.

Task 3

1. List all the classes in the ontology on a sheet of paper and assign your own meta-properties.

2. Compare your assignments with the ones provided in Figure 4.

3. Now add the metaproperty assignments to the OWL file, be it either your own assignments or the ones provided in Figure 4: for each Individual, declare it an instance of the respective metaproperty class. For instance, the instance version of *Paper* has to become an instance of `ontoclean:DependentClass`.

We provide a sample of meta-property assignments for all terms in this tutorial as shown in Figure 4. The file `OntocleanTutorialOntologyPunnedMetaProperties.owl` contains all the assignments per individual. You are encouraged to also make your own separate assignment to the ontology by completing Task 3. Note that your meta-property assignments may be different because the "same [terms from an ontology represent] different concepts to different people" [Welty, 2006]. If your meta-property assignments are different from Figure 4, you are encouraged to create your own version of the `OntocleanTutorialOntologyPunnedMetaProperties.owl` file. The next section will show you how to detect inconsistencies in the provided (or your own) punned file with assigned meta-properties.

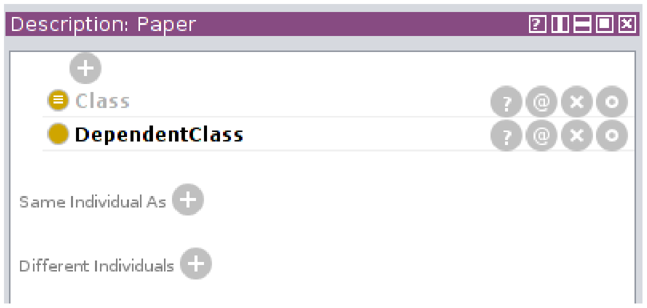


Figure 3: *Paper* individual meta-property assignment.

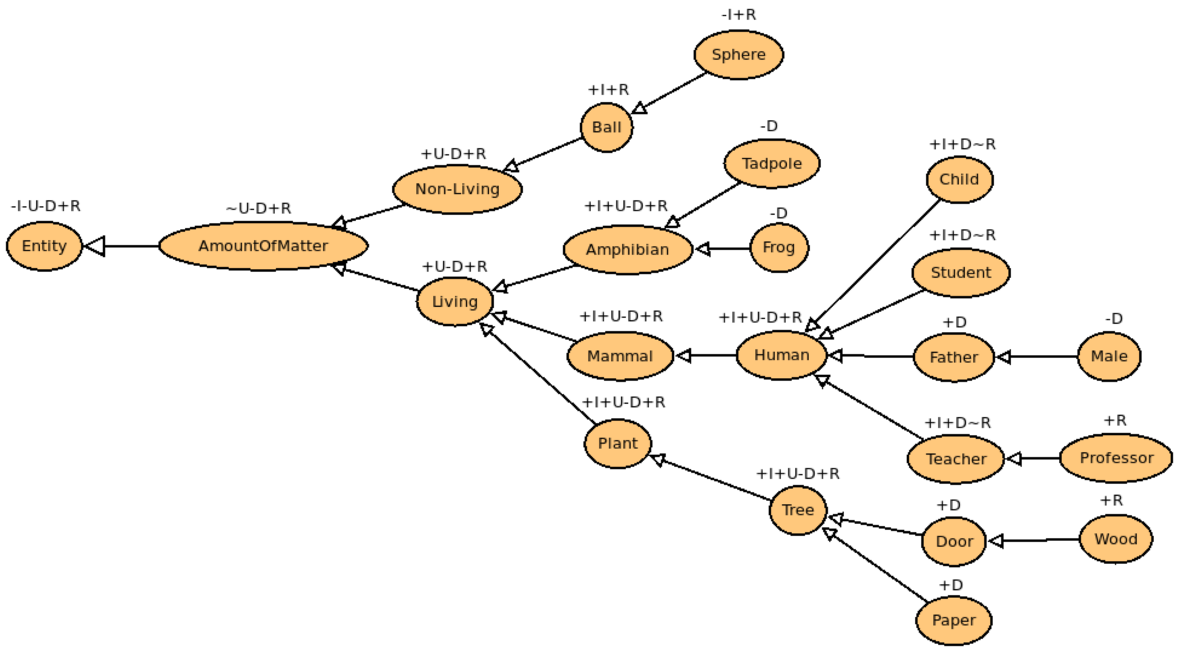


Figure 4: Tutorial ontology with possible meta-properties assigned.

2.5 Detecting inconsistencies in the hierarchy

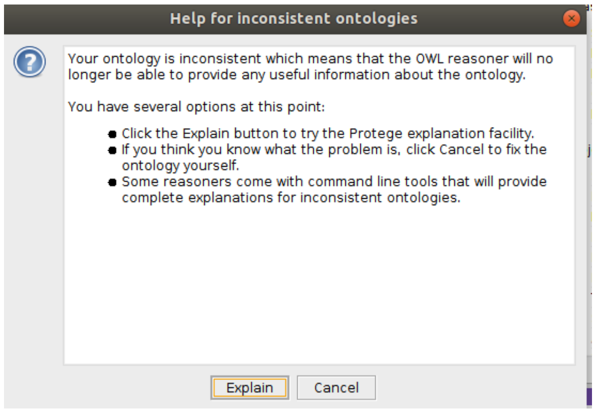
Once meta-properties have been assigned and added to the OWL file, inconsistencies can be discovered by starting a reasoner[4]. For instance, when *HermiT 1.3.8* is started on the provided `OntocleanTutorialOntologyPunnedMetaProperties.owl` file, you should encounter something that looks like the screenshot in Figure 5. Contrary to intended use of reasoners for ontology development, this is supposed to happen. Upon clicking the explain button then a pop-up with a list of inconsistencies will be shown (see Figure 6). Carry out Task 5.

A number of errors should be detected, be it either form your own assignments or the one provided. In the remainder of this section, we describe some of those errors and their causes.

AmountOfMatter and Living

AmountOfMatter and Living have an inconsistency and violate the unity rule that says an entity with a unity tag may not be a subclass of an anti-unity entity. In this case *AmountOfMatter* has an anti-unity tag as amount of matter may be any object but may not be considered as a whole (for instance, water) hence there can be no one unifying criterion. The Living entity has a unifying criterion, that is, any instance of it should be living though it may be any living object hence it may not be a subclass of *AmountOfMatter*.

[4]`https://protegewiki.stanford.edu/wiki/Using_Reasoners`

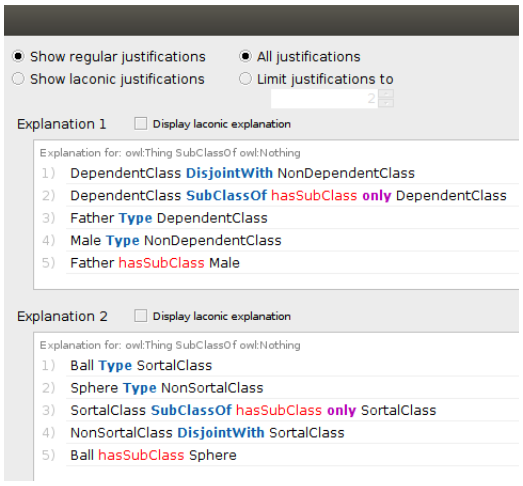

Help for inconsistent ontologies

Your ontology is inconsistent which means that the OWL reasoner will no longer be able to provide any useful information about the ontology.

You have several options at this point:

- Click the Explain button to try the Protege explanation facility.
- If you think you know what the problem is, click Cancel to fix the ontology yourself.
- Some reasoners come with command line tools that will provide complete explanations for inconsistent ontologies.

Figure 5: Reasoner notification for inconsistencies.


● Show regular justifications ○ Show laconic justifications

● All justifications ○ Limit justifications to 2

Explanation 1 ☐ Display laconic explanation

Explanation for: owl:Thing SubClassOf owl:Nothing

1) DependentClass DisjointWith NonDependentClass
2) DependentClass SubClassOf hasSubClass only DependentClass
3) Father Type DependentClass
4) Male Type NonDependentClass
5) Father hasSubClass Male

Explanation 2 ☐ Display laconic explanation

Explanation for: owl:Thing SubClassOf owl:Nothing

1) Ball Type SortalClass
2) Sphere Type NonSortalClass
3) SortalClass SubClassOf hasSubClass only SortalClass
4) NonSortalClass DisjointWith SortalClass
5) Ball hasSubClass Sphere

Figure 6: Section of detected inconsistencies.

Task 5

1. Ensure to have open either the provided OWL file that has the meta-properties assigned (`OntocleanTutorialOntologyPunnedMetaProperties.owl`) or your own one where the meta-properties have been assigned.

2. Start the reasoner.

3. Click the "Explain" button. You can either let it compute as many explanations as it will, or stop after a few. We assume you stop after a few.

4. Examine the first explanation for the inconsistency. What is the cause of it? That is: which OntoClean rule did it violate?

5. Once you are convinced which rules it violated, correct the taxonomy in the TBox (i.e., among the classes) and their corresponding individuals in the ABox. Resolution can be either to change the meta-property assignment or to change the position of the classes in the taxonomy.

6. Carry out steps 2-5 until there are no reported inconsistencies.

AmountOfMatter and Amphibian

Amphibian and *AmountOfMatter* are also inconsistent as they violate the unity rule that states that entities that possess positive unity tags may not be subclasses of entities with anti-unity tags. Though not a direct subclass of *AmountOfMatter*, *Amphibian* is however indirectly a subclass through the relation to the Living entity which is a subclass of *AmountOfMatter* and automatically makes it a subclass of *AmountOfMatter* too.

Sphere and Ball

Sphere and *Ball* also have an inconsistency as they violate the identity constraint that says that an entity with an identity criterion may not be a super class to a non-identity class. A ball has an identifying criterion in that a ball may easily be recognised as a ball, whereas a sphere may be any object that has a sphere shape.

Father and Male

Father and *Male* entity relationship violates the dependence constraints which states that a non-dependent entity may not be a subclass of a dependent class. Thus, for someone to be a male they do not need to be father, in other words Male does not depend on *Father*. Rather, conventionally for you to be a father then you must be a male.

Door and Wood

The *Door* and *Wood* relationship is not correct and is inconsistent in that it violates the rigidity rule. *Door* cannot be a super class of wood since it is anti-rigid and *Wood* is rigid. A door is considered anti-rigid in the sense that a door may cease to be a door say, when it breaks and get replaced. Whereas wood will always be wood whether it is used to make some furniture or just stored for future use.

Teacher and Professor

The relationship between *Teacher* and *Professor* violates the rigidity rule. The professor is a rigid concept as a person once granted the title they will always be a professor as long as they live. A teacher however may cease to be a teacher at any moment or stage in their life, hence it is anti-rigid.

AmountOfMatter and Mammal

The relationship between *AmountOfMatter* and *Mammal* via *Living* is inconsistent as it violates the unity constraint that states that an anti-union class may not be a super class of a unity class. Mammals have unifying characteristics that may include giving birth to live babies and that they are all vertebrates but amount of water may not be unified as other matter may not be quantifiable as a whole for example water.

AmountOfMatter and Human

The relationship between *AmountOfMatter* and *Human* violates the unity rule since *AmountOfMatter* is anti-rigid and *Human* is rigid. It is not a direct violation but since *Human*'s super class has an ancestor class (*AmountOfMatter*) that is anti-unity. *AmountOfMatter* may not be considered as a whole for all its instances so they cannot unified but *Human* has a unifying criteria that may include being warm blooded and giving birth to live offspring.

Human and Teacher

The relationship between *Human* and *Teacher* violates the unity rule; a non-unity class may not be a subclass of an anti-unity class.

2.6 Fixing the hierarchy

Following on from Task 5, you should have resolved each inconsistency so that the reasoner does not return you any more errors like in Figure 5. These errors can be fixed by either re-arranging the position of the classes in the taxonomy, or, upon closer inspection, one may have decided to change the meta-property assignment. An example of the former is that your revised hierarchy surely should not have *AmountOfMatter* at the top. An example of the latter is that a +R on *Professor* would generally be considered to be incorrect[5], but is ∼R instead.

References

[Guarino and Welty, 2000a] Guarino, N. and Welty, C. (2000a). A formal ontology of properties. In Dieng, R. and Corby, O., editors, *Proceedings of 12th International Conference on Knowledge Engineering and Knowledge Management (EKAW'00)*, volume 1937 of *LNCS*, pages 97–112. Springer Verlag.

[Guarino and Welty, 2000b] Guarino, N. and Welty, C. (2000b). Identity, unity, and individuality: towards a formal toolkit for ontological analysis. In Horn, W., editor, *Proceedings of ECAI'00*, pages 219–223. IOS Press, Amsterdam.

[Guarino and Welty, 2009] Guarino, N. and Welty, C. A. (2009). An overview of OntoClean. In Staab, S. and Studer, R., editors, *Handbook on Ontologies*, International Handbooks on Information Systems, pages 201–220. Springer.

[Keet, 2018] Keet, C. M. (2018). *An introduction to ontology engineering.* University of Cape Town.

[Welty, 2006] Welty, C. A. (2006). OntOWLClean: Cleaning OWL ontologies with OWL. In Bennett, B. and Fellbaum, C., editors, *Formal Ontology in Information Systems, Proceedings of the Fourth International Conference, FOIS 2006, Baltimore, Maryland, USA, November 9-11, 2006*, volume 150 of *Frontiers in Artificial Intelligence and Applications*, pages 347–359. IOS Press.

[5]being a professor is a role that a human plays, so it would be an anti-rigid property, and then not violating the rigidity rule anymore.