# WHY DOES THE OWL REASONER IGNORE MY CONSTRAINT?

HENRIETTE HARMSE

A most frustrating problem often encountered by people, with experience in relational databases when they are introduced to OWL ontologies, is that OWL ontology reasoners seem to ignore constraints. In this post I give examples of this problem, explain why they happen and I provide ways to deal with each example.

## 1. An Example

A typical example encountered in relational databases is that of modeling orders with orderlines, which can be modeled via `Orders` and `Orderlines` tables where the `Orderlines` table has a foreign key constraint to the `Orders` table. A related OWL ontology is given in Figure 1. It creates as expected `Order` and `Orderline` classes with a `hasOrder` object property. That individuals of `Orderline` are necessarily associated with **one** order is enforced by `Orderline` being a subclass of `hasOrder exactly 1 owl:Thing`.

## 2. Two Problems

Two frustrating and most surprising errors given the Order ontology are: (1) if an `Orderline` individual is created for which no associated `Order` individual exists, the reasoner will not give an inconsistency, and (2) if an `Orderline` individual is created for which two or more `Order` individuals exist, the reasoner will also not give an inconsistency.

---

*Date*: 10th March 2018.

```
ObjectProperty: hasOrder
    Domain:
        Orderline
    Range:
        Order

Class: Order

Class: Orderline
    SubClassOf:
        hasOrder exactly 1 owl:Thing

Class: owl:Thing
```

Figure 1. An ontology modeling orders and orderlines

```
Individual: orderline123
    Types:
        Orderline,
        hasOrder max 0 owl:Thing
```

FIGURE 2. Enforcing that `orderline123` is not associated via `hasOrder`

```
Individual: order1
    Types:
        Order

Individual: order2
    Types:
        Order

Individual: orderline123
    Types:
        Orderline
    Facts:
     hasOrder  order1,
     hasOrder  order2
```

FIGURE 3. `orderline123` associated with two individuals of type `Order`

2.1. **Missing Association Problem.** Say we create an individual `orderline123` of type `Orderline`, which is not associated with an individual of type `Order`, in this case the reasoner will not give an inconsistency. The reason for this is due to the **open world assumption**. Informally it means that the only inferences that the reasoner can make from an ontology is based on explicit information stated in the ontology or what can derived from explicit stated information.

When you state `orderline123` is an `Orderline`, there is **no explicit information** in the ontology that states that `orderline123` is **not** associated with an individual of `Order` via the `hasOrder` property. To make explicit that `orderline123` is not in such a relation, you have to define `orderline123` as in Figure 2. `hasOrder max 0 owl:Thing` states that it is known that `orderline123` is not associated with an individual via the `hasOrder` property.

2.2. **Too Many Associated Individuals Problem.** Assume we now change our definition of our `orderline123` individual to be associated via `hasOrder` to two individuals of `Order` as shown in Figure 3. Again, most frustratingly the reasoner does not find that the ontology is inconsistent. The reason for this is that OWL does not make the **unique name assumption**. This means that individuals with different names can be assumed by the reasoner to represent a single individual. To force the reasoner to see `order1` and `order2` as necessarily different, you can state `order1` different from `order2` by adding `DifferentFrom:order2` to `order1` (or similarly for `order2`).

## 3. Constraint Checking versus Deriving Inferences

The source of the problems described here is due to the difference between the purposes of a relational database and an OWL reasoner. The main purpose of a relational database is to enable view and edit access of the data in such a way that the integrity of the data is maintained. A relational database will ensure that

the data adheres to the constraints of its schema, but it cannot make any claims beyond what is stated by the data it contains. The main purpose of an OWL reasoner is to derive inferences from statements and facts. As an example, from the statement `Class: Dog SubclassOf: Animal` and the fact `Individual: pluto Type: Dog` it can be derived that `pluto` is an `Animal`, even though the ontology nowhere states explicitly that `pluto` is an `Animal`.

## 4. Conclusion

Many newcomers to OWL ontologies get tripped up by the difference in purpose of relational databases and OWL ontologies. In this post I explained some of the challenges that stem from this difference in purpose and how to deal with them.

## References