The Minimal Finite Model Visualization as an Ontology Debugging Tool

Guntis Barzdins and Martins Barinskis

IMCS, University of Latvia
martins.barinskis@gmail.com, guntis.barzdins@mii.lu.lv

Abstract. We present an new *Protégé* plugin for constructing a minimal satisfiability model of an OWL ontology and visualizing it in the original music score notation.

1 Introduction

The ontology satisfiability is a property that indicates whether all classes defined in the ontology can possibly have any instances. Satisfiability implies consistency. The described *Protégé* plugin [1] complements the traditional ontology satisfiability debugging tools [2]: if an automatically constructed minimal model of the ontology contradicts the author's intentions, then the ontology itself is either wrong or incomplete.

Currently, we consider only ontologies with a finite satisfiability model. Here we focus on the model visualization enhancements, while the overall approach is described in [3].

2 Acquiring and Visualizing a Satisfiability Model

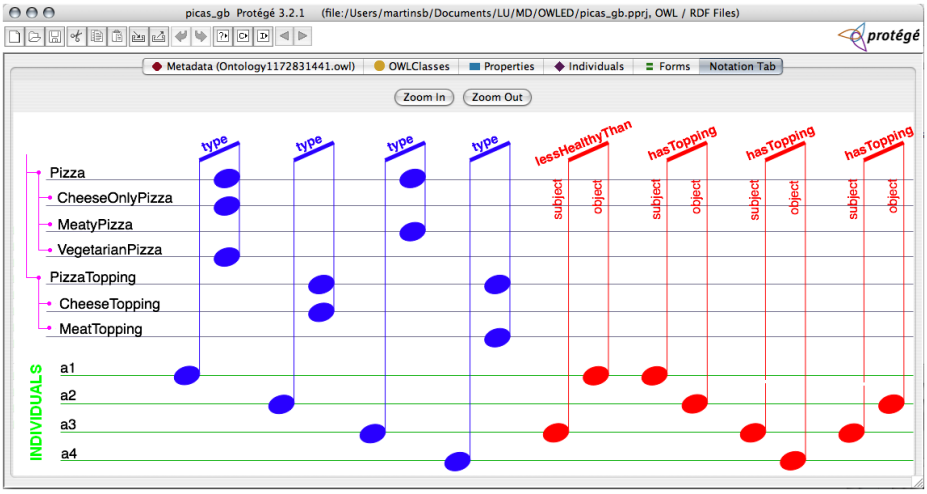
Let us consider a simple OWL DL pizza ontology:

```
Ontology(
    Class(Pizza partial restriction(hasTopping
        someValuesFrom(PizzaTopping)) owl:Thing)
    Class(MeatyPizza complete restriction(hasTopping
        someValuesFrom(MeatTopping)) Pizza
        restriction(lessHealthyThan someValuesFrom(VegetarianPizza)))
    Class(CheeseOnlyPizza complete restriction(hasTopping
        someValuesFrom(CheeseTopping)) Pizza
        restriction(hasTopping allValuesFrom(CheeseTopping)))
    Class(VegetarianPizza complete complementOf(restriction(hasTopping
        someValuesFrom(MeatTopping))) Pizza)
    Class(PizzaTopping partial owl:Thing)
    Class(CheeseTopping partial PizzaTopping)
    Class(MeatTopping partial PizzaTopping)
    DisjointClasses(CheeseTopping MeatTopping)
    DisjointClasses(PizzaTopping Pizza)
    ObjectProperty(hasTopping domain(Pizza) range(PizzaTopping))
)
```

Pellet and Fact++, the popular OWL DL/OWL 1.1 reasoners, do not output an ontology satisfiability model. Therefore, to build a model, we map the OWL ontology to the FOL formula and use *Mace4*, a FOL model builder:

```
relation(CheeseOnlyPizza(_), [ 1, 0, 0, 0 ]),
relation(CheeseTopping(_), [ 0, 1, 0, 0 ]),
relation(MeatTopping(_), [ 0, 0, 0, 1 ]),
relation(MeatyPizza(_), [ 0, 0, 1, 0 ]),
relation(Pizza(_), [ 1, 0, 1, 0 ]),
relation(PizzaTopping(_), [ 0, 1, 0, 1 ]),
relation(VegetarianPizza(_), [ 1, 0, 0, 0 ]),
relation(hasTopping(_,_), [ 0, 1, 0, 0,
                            0, 0, 0, 0,
                            0, 1, 0, 1,
                            0, 0, 0, 0 ]),
relation(lessHealthyThan(_,_), [ 0, 0, 0, 0,
                            0, 0, 0, 0,
                            1, 0, 0, 0,
                            0, 0, 0, 0 ])
```

The created satisfiability model is further visualized using an original "music score notation":



The traditional class hierarchy and individuals are visualized as lines that are interconnected by the notes (relations). Filtering can be used to navigate larger models.

References

1. Barinskis, M., Barzdins, G.: OWL DL Satisfiability Model Visualization Plugin for Protege: http://apps.lumii.lv/satmodviz/
2. Parsia, B., Sirin, E., Kalyanpur, A.: Debugging OWL Ontologies. In: WWW'05, ACM Press (2005) 633–640
3. Barinskis, M., Barzdins, G.: Satisfiability Model Visualization Plugin for Deep Consistency Checking of OWL Ontologies. In: OWLED 2007 (to be published). (2007)