Debugging of $\mathcal{ALC}$-Ontologies via Minimal Model Generation

Fabio Papacchini  Renate A. Schmidt

School of Computer Science
The University of Manchester

April 9, 2015
What Ontologies are

Basis for semantic web and knowledge-based systems

Widely used in practice: BBC, NHS, Klappo, . . .

A formalism for knowledge representation

- defining terminology for the domain of interest
- based on description logic
- allowing for automated reasoning techniques
What Ontologies are – Example

**Pizza Ontology**

\[ \forall \text{hasTopping}. \text{VegetarianTopping} \sqsubseteq \text{VegetarianPizza} \]

\[ \text{MozzarellaTopping} \sqsubseteq \text{VegetarianTopping} \]

\[ \text{TomatoTopping} \sqsubseteq \text{VegetarianTopping} \]

\[ \text{Margherita} \sqsubseteq \forall \text{hasTopping}.(\text{MozzarellaTopping} \sqcup \text{TomatoTopping}) \]

\[ \text{Margherita} \sqsubseteq \exists \text{hasTopping}.\text{MozzarellaTopping} \]

\[ \text{Margherita} \sqsubseteq \exists \text{hasTopping}.\text{TomatoTopping} \]

\[ \vdash \]

Reasoning on the ontology allows to derive implicit information such as

\[ \text{Margherita} \sqsubseteq \text{VegetarianPizza} \]
Ontology Debugging

Someone has to model the domain of interest.

The resulting ontology is supposed to

- be coherent (no concept is unsatisfiable)
- model properly (implicit) domain knowledge

Ontology debugging aims to guarantee that an ontology

- has these properties
- keeps these properties when modified
Underspecification

Pizza Ontology

\[ \forall \text{hasTopping}. \text{VegetarianTopping} \sqsubseteq \text{VegetarianPizza} \]

\[ \text{MozzarellaTopping} \sqsubseteq \text{VegetarianTopping} \]

\[ \text{TomatoTopping} \sqsubseteq \text{VegetarianTopping} \]

\[ \text{Margherita} \sqsubseteq \forall \text{hasTopping}. (\text{MozzarellaTopping} \sqcup \text{TomatoTopping}) \]

\[ \text{Margherita} \sqsubseteq \exists \text{hasTopping}. \text{MozzarellaTopping} \]

\[ \text{Margherita} \sqsubseteq \exists \text{hasTopping}. \text{TomatoTopping} \]

\[ \vdots \]

It is no longer true that \text{Margherita} \sqsubseteq \text{VegetarianPizza}. 
Our Approach

Use model generation as tests in test-driven software development.

Given an ontology $\mathcal{O}$ and a set $S_\alpha$ of properties, check if $\mathcal{O} \models \alpha$ 
$(\mathcal{O} \cup \{\neg \alpha\} \not\models \bot)$ for all $\alpha \in S_\alpha$.

- if $\mathcal{O} \not\models \alpha$
  - extraction of a model where $\neg \alpha$ is true
  - the model is an explanation of why $\mathcal{O} \not\models \alpha$
  - understanding the model allows to fix the ontology
- if $\mathcal{O} \models \alpha$ then $\mathcal{O}$ is well specified w.r.t. $\alpha$

This approach can be used at any stage of the life cycle of an ontology.
Pizza Ontology

∀hasTopping. VegetarianTopping ⊆ VegetarianPizza

MozzarellaTopping ⊆ VegetarianTopping

TomatoTopping ⊆ VegetarianTopping

Margherita ⊆ ∀hasTopping. (MozzarellaTopping ∪ TomatoTopping)

Margherita ⊆ ∃hasTopping. MozzarellaTopping

Margherita ⊆ ∃hasTopping. TomatoTopping

\ldots

Check \( \mathcal{O} \cup \{(\text{Margherita} \cap \neg \text{VegetarianPizza})(a)\} \models \bot \)

\{\text{Mozzarella, VegetarianTopping}\}

\{\text{Margherita}\} \rightarrow \{\text{TomatoTopping}\}