UNIVERSITY OF AMSTERDAM

Analysis of a description logic ontology

Simonetto Luca - 11413522
Ruben Blom - 10684980
Knowledge Representation

October 10, 2016

Task 1: *find which classes are inconsistent, find ways to resolve the inconsistency, implement them in the Protege editor*

For this and the following tasks the reasoner used was FaCT++.

After loading the pizza.owl ontology in the Protege editor and starting the reasoner, we discovered two separate inconsistencies, so we took two separate approached to resolve them.

CHEESYVEGETABLETOPPING

- Problem: `CheesyVegetableTopping` is subclass of `CheeseTopping` and `VegetableTopping`, thus creating a conflict as the two superclasses are disjoint.
- Analysis: in essence, there is not a topping that is at the same time a kind of cheese and a vegetable, so a possible solution could be to simply delete it. A food with

  `hasTopping some CheesyVegetableTopping` would be written as

  `(hasTopping some CheesyTopping) and (hasTopping some VegetableTopping)`

  We assume that no topping can be a cheese and a vegetable at the same time.
- Solution: remove the class `CheesyVegetableTopping`.

ICECREAM

- Problem: The `IceCream` class has as subclass `hasTopping some FruitTopping` and the propriety `hasTopping` has only `Pizza` domain. This conflict requires a more elaborated solution.

- Analysis: We have encountered a number of different subproblems, as the propriety `hasTopping` has an incompatible domain and the `PizzaTopping` class has now a wrong meaning (too restrictive).
- Solution: we first need to solve the `hasTopping` problem, giving the domain Food. Then we also have to fix the reverse propriety `isToppingOf`, as the `Range` propriety has also to range on `Food`. The next step we have done is to remove the propriety `hasTopping some FruitTopping` from `IceCream` (in order to make it similar to the `Pizza` class) and adding the subclass `FruityIceCream`, which is equivalent to `IceCream and (hasTopping some FruitTopping)`. Now, as `FruityIceCream` would have a `PizzaTopping`, we renamed `PizzaTopping` to `Topping`.

Task 2: *find if there are places where the pizza.owl ontology contains redundant statements. If you find any, report them and explain why they are redundant.*

While searching through the whole ontology we discovered two types of redundancies: equivalent classes and unnecessary subclass declarations.

- `SpicyPizzaEquivalent` ≡ `SpicyPizza`: the first is equivalent to `Pizza and (hasTopping some (PizzaTopping and (hasSpiciness some Hot)))` while the second is equivalent to `Pizza and (hasTopping some SpicyTopping)`. The two declarations are in fact equivalent as we can see from the `SpicyTopping` definition.
- `VegetarianPizzaEquivalent2` ≡ `VegetarianPizzaEquivalent1`: we can easily spot it as the range of available toppings of `VegetarianPizzaEquivalent2` is in fact the definition of the available toppings for `VegetarianPizzaEquivalent1`. [Sidenote] we also fixed the class `VegetarianPizza`, as we think that removing meat and fish from the possible toppings would leave a new kind of topping to be part of a `VegetarianPizza`. In the case that a new kind of topping is introduced, if it is considered vegetarian it will have to be inserted in the list of vegetarian toppings.
- `SlicedTomatoTopping` and `SundriedTomatoTopping` subclasses: in this case we can see that the declaration `SubClass Of` of the two classes is an unnecessary repetition. The value `hasSpiciness some Mild` is also declared on the `TomatoTopping` superclass, so it can be removed.

Task 3: *Extend the ontology with some obvious redundant statements.*

We added three redundancies in the ontology, explained below

- In the class `RedOnionTopping` we have added the subclass `hasSpiciness some Medium`, that is redundant with the declaration of the parent class.
- We added the class `HotPizza`, equivalent to `Pizza and (hasTopping some SpicyTopping)`, making it equivalent to `SpicyPizza`.
- For the pizza `Siciliana` we added the propriety `SubClass Of Food`, that is useless as it is already a child class of it.