1 OWL Axiom for Human

Assume an ontology that contains (at least) the class Human and its two subclasses Man and Woman. Assume two further properties hasFather and hasMother.

1. Define an OWL axiom (p. 39) that *fully* characterizes Human, viz., that it has *exactly one* father of type Man and one mother of type Woman. In order to achieve this, you need different OWL class constructors (p. 36).

2. Together with the non-unique name assumption (p. 40), functional properties as those above sometimes lead to surprising inferences. Give a small example using the above ontology!

Solution 1. $\mathsf{Man} \sqsubseteq \mathsf{Human}$
$\mathsf{Woman} \sqsubseteq \mathsf{Human}$
$\mathsf{Human} \equiv \leq 1.\mathsf{hasFather} \sqcap \geq 1.\mathsf{hasFather} \sqcap$
$\leq 1.\mathsf{hasMother} \sqcap \geq 1.\mathsf{hasMother} \sqcap$
$\exists \mathsf{hasFather}.\mathsf{Man} \sqcap \exists \mathsf{hasMother}.\mathsf{Woman}$

Note: it does NOT matter whether we are using universal or existential property restrictions in the third line.

Solution 2. Assume that the ABox of the above ontology would contain the two relation instances (functional notation) hasMother(peter, mary) and hasMother(peter, marian). Functionality of the hasMother relation then requires that mary and marian are equal: owl:sameAs(mary, marian).

2 rdfs:subClassOf and owl:intersectionOf

Is the denotation of Z in ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$ the same for every interpretation, assuming X and Y have the same denotation in $\mathcal{O}_1$ and $\mathcal{O}_2$?

$\mathcal{O}_1 = \{\mathsf{Z} \equiv \mathsf{X} \sqcap \mathsf{Y}\}$
$\mathcal{O}_2 = \{\mathsf{Z} \sqsubseteq \mathsf{X}, \mathsf{Z} \sqsubseteq \mathsf{Y}\}$

If so, why? If not so, why? Hint: you can use Venn diagrams to explain your choice.

Solution. There exists an interpretation $\mathcal{I}$, where all individuals of Z in $\mathcal{O}_2$ belong to only a *subset* of $\mathsf{X} \sqcap \mathsf{Y}$, meaning that there exists an *exhaustive disjoint partition* p of $\mathsf{X} \sqcap \mathsf{Y}$, i.e.,

$$p = \mathsf{Z}^{\mathcal{I}} \cup \mathsf{W}^{\mathcal{I}} \textbf{ and } \mathsf{Z}^{\mathcal{I}} \cap \mathsf{W}^{\mathcal{I}} = \emptyset$$

This is not possible in $\mathcal{O}_1$, since Z is exactly ($\equiv$) defined through $\mathsf{X} \sqcap \mathsf{Y}$.

3 Integrity Constraints

Certain relations (e.g., spatio or temporal topological relations) are often anti-reflexive. For instance,

partOf(s,t) $\rightarrow \neg$partOf(t, s) e.g., partOf(door, house)
before(s,t) $\rightarrow \neg$before(t, s) e.g., before(5am, 5pm)

1. You can NOT write such a rule in SWRL/OWLIM, since negation is restricted to OWL class constructors in the TBox. However, you can rewrite those rules with negation on the RHS such that the new rule essentially captures the semantics of the old rule. This however requires querying for individuals of a very special class after the application of the rewritten rule.
2. Would it be possible to have negation on the LHS of a rule. Would this lead to problems? Explain!
3. Negation on the LHS and the RHS is allowed in SWRL/OWLIM within a specific axiom. Which one?

Solution 1. Rewrite $\alpha \rightarrow \neg\beta$ as $\alpha \wedge \beta \rightarrow \bot$ and query the resulting ontology for individuals of type owl:Nothing:

$$\alpha \rightarrow \neg\beta \Longleftrightarrow \neg\alpha \vee \neg\beta \Longleftrightarrow \neg(\alpha \wedge \beta) \Longleftrightarrow \neg(\alpha \wedge \beta) \vee \bot \Longleftrightarrow \alpha \wedge \beta \rightarrow \bot$$

Solution 2. The order-independence (free choice) of rules from R (p. 52) in the forward chaining algorithm is no longer guaranteed and will result into different result sets T.
Consider the two rules $(R : \neg\alpha \rightarrow \beta)$ and $(R' : \gamma \rightarrow \alpha)$. $\beta \in T$ if R is applied before R'. Assuming that only R is able to add β to T and that γ has a matching, $\beta \notin T$ iff R' is applied before R.

Solution 3. The axiom uses differentFrom which explicitly states that two *individuals* can NEVER be equal.

4 Forward Chaining

Why does the forward chaining algorithm (p. 52) terminate, assuming we are dealing with *safe* entailment rules (p. 48) ?

Solution. Neither new URIs nor new XSD literals are introduced during the matching & instantion phase. Only new triples are constructed during instantiation when variables in the head of a rule are replaced by values from the binding environment.
Assuming that n = #URIs + #XSD literals, we will have at most n^3 triples at the very end of the closure computation. This is a finite number, hence the fixpoint will be reached in a finite number of iterations.
The deeper reason for having a finite fixpoint ($T = T'$) lies in the nature of set union: $\cup$ is a monotonic operation that will not delete information from T.

Furthermore and also very important: we normally refer to the Cantor notion of sets—sets must not contain duplicate elements.

5 Reducibility to Consistency

On page 41, we said that the basic inference problems are all reducible to (in)consistency. Reformulate subsumption: $\mathsf{C} \sqsubseteq \mathsf{D}$. You might again use Venn diagrams to explain your reformulation.

Solution. $\mathsf{C} \sqsubseteq \mathsf{D} \iff \mathsf{C} \sqcap \mathsf{D} \equiv C \iff \mathsf{C} \sqcap \mathsf{D} \sqcap \neg\mathsf{D} \equiv \mathsf{C} \sqcap \neg\mathsf{D} \iff \mathsf{C} \sqcap \neg\mathsf{D} \equiv \bot$, i.e., C subsumes D iff $\mathsf{C} \sqcap \neg\mathsf{D}$ is inconsistent.

6 OWL Entailment Rules

Page 56 has presented four OWL entailment rules in OWLIM notation. Now, properties in OWL can either be (see pp. 34, 36, 37)

- *datatype properties* of rdf:type owl:DatatypeProperty
  example: hasAge(peter, 42)) or
- *object properties* of rdf:type owl:ObjectProperty
  example: hasMother(peter, mary)).

Orthogonal to this, properties in OWL are furthermore classified as being transitive, symmetric, *functional*, etc. (p. 35). Write OWL entailment rules that address functional object as well as functional datatype properties (you will probably need inequality constraints over variables; see p. 57). Are these rules different. Why?

Solution.

```
p <rdf:type> <owl:FunctionalProperty>
p <rdf:type> <owl:DatatypeProperty>
i p j
i p k        [Constraint j != k]
-------------------------------------
i <rdf:type> <owl:Nothing>
```

```
p <rdf:type> <owl:FunctionalProperty>
p <rdf:type> <owl:ObjectProperty>
i p j
i p k        [Constraint j != k]
-------------------------------------
j <owl:sameAs> k
```

Different XSD literals, such as 42 and 2 can NEVER be the same, i.e., values in the range of a datatype property are only equal if they are literally the same. However, due to the non-unique name assumption, different URIs might refer to the same individual in the range of an object property.