Inference in Description Logic $\mathcal{ALC}$

Petr Křemen

December 14, 2017

1 Inference Procedures

Ex. 1 — Why inconsistency of an ontology is a problem ? What is its consequence ?

Answer (Ex. 1) — The logical calculus of $\mathcal{ALC}$ is based on first order logic. Thus, an inconsistent ontology entails all axioms.

Ex. 2 — Show that disjointness of two concepts can be reduced to unsatisfiability of a single concept.

Answer (Ex. 2) — Let's reproduce the flow of equivalent operations for this simple transformation:

$$\mathcal{K} \models C \sqsubseteq \neg D \tag{1}$$
$$(\forall \mathcal{I})(\mathcal{I} \models \mathcal{K}) \Rightarrow (\mathcal{I} \models (C \sqsubseteq \neg D)) \tag{2}$$
$$(\forall \mathcal{I})(\mathcal{I} \models \mathcal{K}) \Rightarrow (C^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \setminus D^{\mathcal{I}})) \tag{3}$$
$$(\forall \mathcal{I})(\mathcal{I} \models \mathcal{K}) \Rightarrow (C^{\mathcal{I}} \cap D^{\mathcal{I}} \subseteq (\Delta^{\mathcal{I}} \setminus D^{\mathcal{I}}) \cap D^{\mathcal{I}} = \{\})) \tag{4}$$
$$(\forall \mathcal{I})(\mathcal{I} \models \mathcal{K}) \Rightarrow (\mathcal{I} \models (C \sqcap D \sqsubseteq \bot)) \tag{5}$$
$$\mathcal{K} \models C \sqcap D \sqsubseteq \bot \tag{6}$$
$$\mathcal{K} \models (C \sqcap D) \text{ is unsatisfiable} \tag{7}$$

Ex. 3 — A concept C is satisfiable w.r.t. $\mathcal{K}$ iff it is interpreted as a non-empty set in at least one model of $\mathcal{K}$. Is it possible to find out that C is interpreted as a non-empty set in all models of $\mathcal{K}$?

Answer (Ex. 3) — If $\mathcal{K} \cup (C \sqsubseteq \bot)$ is inconsistent for consistent $\mathcal{K}$, then $C^{\mathcal{I}} \neq \{\}$ for each model $\mathcal{I}$ of $\mathcal{K}$.

2 Tableaux Algorithm for $\mathcal{ALC}$

Ex. 4 — Decide, whether the $\mathcal{ALC}$ concept $\exists hasChild \cdot (Student \sqcap Employee) \sqcap \neg(\exists hasChild \cdot Student \sqcap \exists hasChild \cdot Employee)$ is satisfiable (w.r.t. an empty TBox). Show the run of the tableau algorithm in detail.

Ex. 5 — Decide, whether the theory/ontology $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ is consistent. Show the run of the tableau algorithm in detail.

- $\mathcal{T} = \{\exists hasChild \cdot \top \equiv Parent\}$
- $\mathcal{A} = \{hasChild(JOHN, MARY), Woman(MARY)\}$

Ex. 6 — Decide and show, whether the ontology

$$\mathcal{K}_1 = (\mathcal{T} \cup \{Parent \sqsubseteq \forall hasChild \cdot \neg Woman\}, \mathcal{A})$$

is consistent.

Ex. 7 — Decide and show, whether the ontology

$$\mathcal{K}_2 = (\mathcal{T} \cup \{Parent \sqsubseteq \exists hasChild \cdot Parent\}, \mathcal{A})$$

is consistent.

Answer (Ex. 7) — To check the consistency, we will use the tableau algorithm for $\mathcal{ALC}$. To keep description compact, we shorten $Parent$,$hasChild$, $Woman$ as P, h, W First, we need to internalize the TBOX

$$\begin{aligned}\{\exists h \cdot \top &\equiv P,\\ P &\sqsubseteq \exists h \cdot P\}\end{aligned}$$

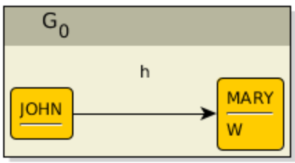
into the single axiom $\top \sqsubseteq \top_C$, such that $\top_C$ is:

$$(\neg(\exists h \cdot \top) \sqcup P) \sqcap (\neg P \sqcup \exists h \cdot \top) \sqcap (\neg P \sqcup \exists h \cdot P) \tag{8}$$

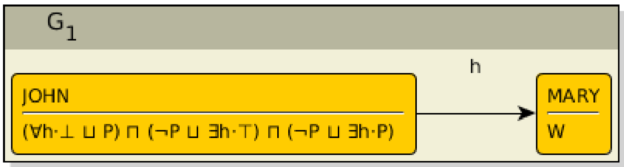
Now, we transform all concepts in $\mathcal{K}_2$ (here only $\top_C$) into negational normal form. T_C :

$$(\forall h \cdot \bot \sqcup P) \sqcap (\neg P \sqcup \exists h \cdot \top) \sqcap (\neg P \sqcup \exists h \cdot P) \tag{9}$$

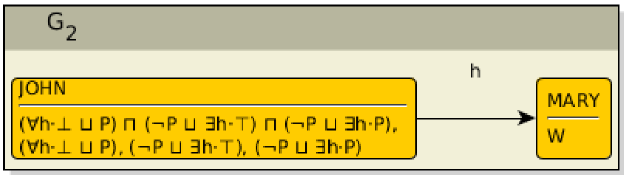
The initial state $S_0 = \{G_0\}$ of the algorithm contains a single *completion graph* G_0 representing the input ABOX



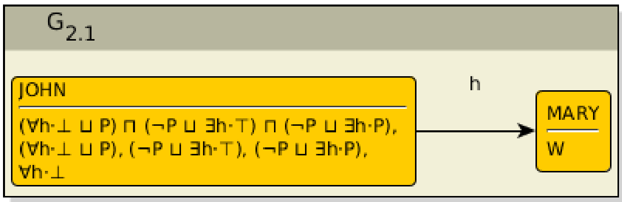
G_0 does not contain a direct clash (there is neither $\bot$, nor A and $\neg A$ in the label of a single node). G_0 is not complete w.r.t $\mathcal{ALC}$ completion rules, as the $\sqsubseteq$ −rule is applicable. Applying the rule on the node $JOHN$ we get a new tableau algorithm state $S_1 = \{G_1\}$ where G_1 is

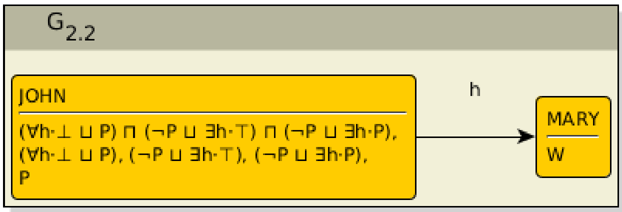


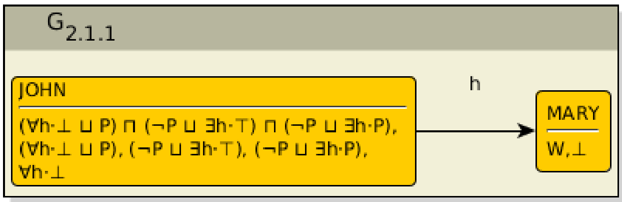
G_1 is clash-free and not complete as well. Two rules are applicable – the $\sqsubseteq -$rule and the $\sqcap-$rule. We apply the latter one (as a heuristic, we expect the clash to be found earlier using the $\sqcap-$ rule) and get the state $S_2 = \{G_2\}$ where G_2 is

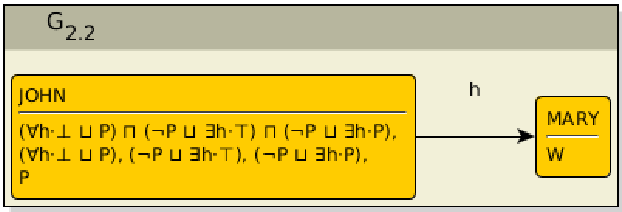


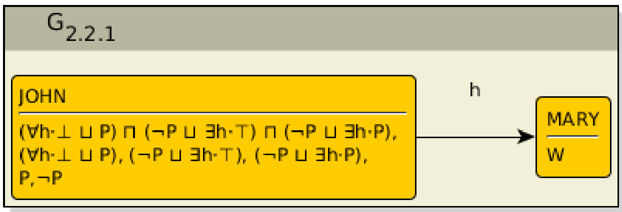
From now on we will proceed more quickly forward and show only tableau reasoner state with the information about rule application and clashing graphs. Whenever more rules are applicable, the one that is applied is marked in green, as well as the chosen graph. Graphs containing a clash are no more shown in the algorithm state.

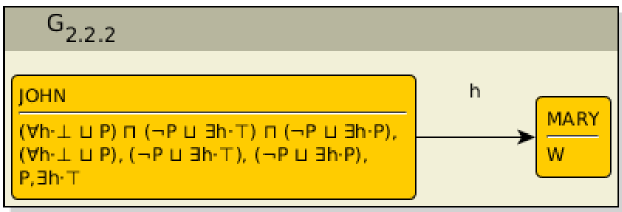
<table>
<tr><th>applicable rules</th><th>state before applying the rule</th><th>state after applying the rule</th></tr>
<tr><td>$\sqsubseteq$, $\sqcup$</td><td>$\{G_2\}$</td><td>




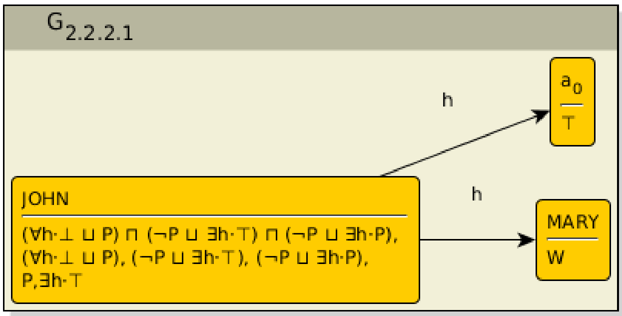
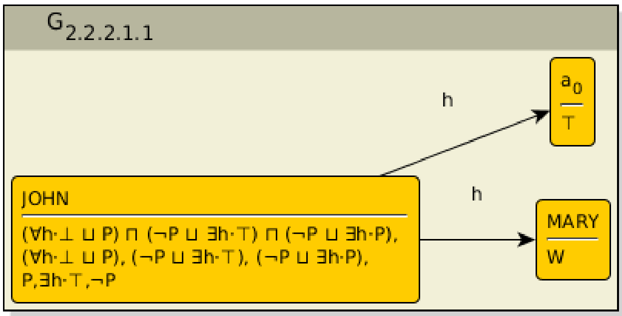
</td></tr>
<tr><td>$\sqsubseteq$, $\sqcup$,$\forall$</td><td>$\{G_{2.1}, G_{2.2}\}$</td><td>


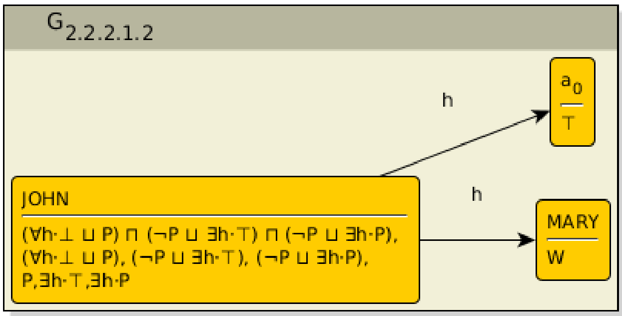


$G_{2.1.1}$ contains a direct clash.
</td></tr>
<tr><td>$\sqsubseteq$, $\sqcup$</td><td>$\{G_{2.2}\}$</td><td>


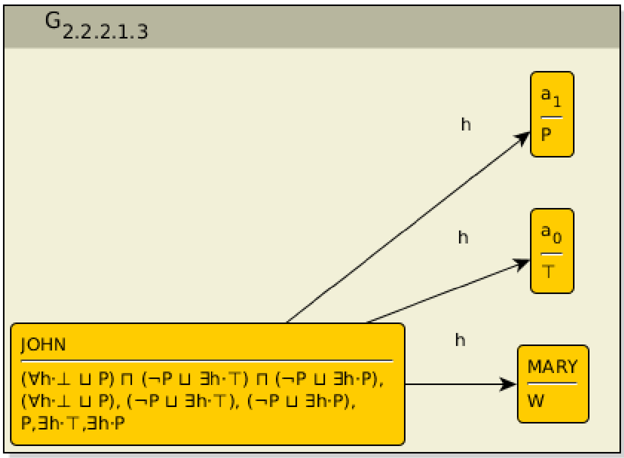


$G_{2.2.1}$ contains a direct clash.
</td></tr>
</table>

<table>
<tr><th>applicable rules</th><th>state before applying the rule</th><th>state after applying the rule</th></tr>
<tr><td>$\sqsubseteq$, $\sqcup$, $\exists$</td><td>$\{G_{2.2.2}\}$</td><td>
</td></tr>
<tr><td>$\sqsubseteq$, $\sqcup$</td><td>$\{G_{2.2.2.1}\}$</td><td>

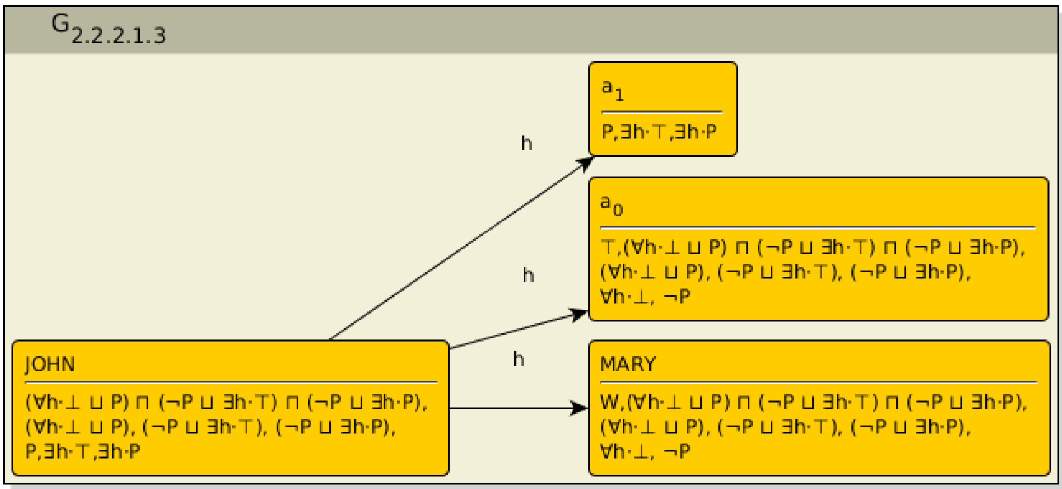


$G_{2.2.2.1.1}$ contains a direct clash.</td></tr>
</table>

<table>
<tr><th>applicable rules</th><th>state before applying the rule</th><th>state after applying the rule</th></tr>
<tr><td>$\sqsubseteq$, $\exists$</td><td>$\{G_{2.2.2.1.2}\}$</td><td>


a_1 is blocked by $JOHN$ as the label of a_0 is a subset of the label of $JOHN$.</td></tr>
</table>

Now, applying the sequence of rules ($\sqsubseteq$,$\sqcup$, $\sqcup$, $\sqcup$) for $MARY$, a_0 and a_1, we get the graph[1]



[1] We do not depict the whole algorithm state, as it contains several graphs of the similar size like G_3 due to the fact that the presence of the label $\forall h \cdot \bot$ in the node a_0 and $MARY$ does lead to a clash, contrary to the case of $JOHN$. This fact generates several alternative disjuncts for each node. Also notice that we chose one set of disjuncts for a_1 and another set of disjuncts for $MARY$ and a_0 in order to avoid clash.

In this graph, a_1 is blocked by $JOHN$ and thus, the $\exists$ rules do note apply. Therefore, this graph is complete and clash-free. The ontology $\mathcal{K}_2$ is consistent.

3 Practically in Protégé

Ex. 8 — Model the previous ontology in Protégé and check (using the Pellet/HermiT reasoner) whether your solutions in the previous tasks were correct.

Ex. 9 — Adjust the Pizza ontology introduced in the previous seminar, so that the class $IceCream$ and $CheeseyVegetableTopping$ become satisfiable.

Ex. 10 — Explain, why the Pizza ontology is consistent, although it contains unsatisfiable classes.