Tableau algorithm for $\mathcal{ALC}$

Petr Křemen

version 0.2

The algorithm, introduced originally in [3] in slighter different form, aims at constructing a model of an $\mathcal{ALC}$ ontology $\mathcal{O} = \mathcal{T} \cup \mathcal{A}$. During the algorithm run, each (possibly infinite) candidate model is represented by a (necessarily finite) *completion graph*.

A *completion graph* is a labeled oriented graph $G = \langle V_G, E_G, L_G \rangle$, where each $x \in V_G$ is labeled with a set $L_G(x)$ of concepts, and each edge $\langle x, y \rangle \in E_G$ is labeled with a set $L_G(\langle x, y \rangle)$ of roles. Furthermore, a completion graph G

- **contains a direct clash**, if $\{\mathsf{A}, \neg\mathsf{A}\} \subseteq L_G(x)$ for some named concept A, or $\bot \in L_G(x)$, or $\neg\top \in L_G(x)$
- **is complete w.r.t. to the set J of rules**, if no completion rule from J can be applied on it.

The tableau algorithm evolves a set $\mathcal{S}$ of completion graphs (corresponding to partial candidate models) according to completion rules in J. Whenever a rule application causes a clash in a completion graph, the graph is discarded (which prunes candidate models corresponding to the graph). The algorithm terminates when no more rules can be applied on a clash-free completion graph (ontology is consistent), or when no completion graph remains to explore (ontology is inconsistent).

1 Tableau algorithm for $\mathcal{ALC}$ with empty TBox

First focus on the case when TBox is empty, i.e. $\mathcal{T} = \emptyset$. In this case the set J of applicable completion rules is depicted in Table 1. The procedure is as follows:

1. (PREPROCESSING) All concepts in $\mathcal{O}$ have to be transformed into Negational Normal Form. This means to "move negation in front of named concepts" using equivalences,like $\neg(C_1 \sqcap C_2) \equiv \neg C_1 \sqcup \neg C_2$, or $\neg(\exists R \cdot C) \equiv \forall R \cdot \neg C$.
2. (INITIALIZATION) Initial state of the algorithm is $\mathcal{S}_0 = \{G_0\}$, where $G_0 = \langle V_{G_0}, E_{G_0}, L_{G_0} \rangle$ is a completion graph, that "corresponds to $\mathcal{A}$", i.e.
   - V_{G_0} contains all named individuals occuring in some axiom of $\mathcal{A}$

- E_{G_0} contains all pairs $\langle \mathsf{a}_1, \mathsf{a}_2 \rangle$ occuring in some $R(\mathsf{a}_1, \mathsf{a}_2) \in \mathcal{A}$.
- L_{G_0} labels each vertex (individual) a with a set $\{C \mid C(\mathsf{a}) \in \mathcal{A}\}$, and each edge $\langle \mathsf{a}_1, \mathsf{a}_2 \rangle$ with a set $\{R \mid R(\mathsf{a}_1, \mathsf{a}_2) \in \mathcal{A}\}$.

3. (CONSISTENCY TEST) Denote the current state as $\mathcal{S}$. Remove from $\mathcal{S}$ any G that *contains a direct clash*. If $\mathcal{S} = \emptyset$ then return INCONSISTENT.

4. (MODEL TEST) Take arbitrary $G \in \mathcal{S}$ that does not contain a direct clash. If G is *complete* with respect to completion rules in J.

5. (RULE APPLICATION) Find a completion rule that is applicable on G. Denote the new state $\mathcal{S}'$ created from the current state $\mathcal{S}$ and go to step 2.

$\sqcap$-**rule**	if:	$(C_1 \sqcap C_2) \in L_G(\mathsf{a})$, for some a and $\{C_1, C_2\} \not\subseteq L_G(\mathsf{a})$.
	then:	$\mathcal{S}' = \mathcal{S} \cup \{G'\} \setminus \{G\}$, where $G' = \langle V_G, E_G, L_{G'} \rangle$.
		$L_{G'} = L_G$ except $L_{G'}(\mathsf{a}) = L_G(\mathsf{a}) \cup \{C_1, C_2\}$.
$\sqcup$-**rule**	if:	$(C_1 \sqcup C_2) \in L_G(\mathsf{a})$, for some a and $\{C_1, C_2\} \cap L_G(\mathsf{a}) = \emptyset$.
	then:	$\mathcal{S}' = \mathcal{S} \cup \{G_1, G_2\} \setminus \{G\}$, where $G_k = \langle V_G, E_G, L_{G_k} \rangle$.
		$L_{G_k} = L_G$ except $L_{G_k}(\mathsf{a}) = L_G(\mathsf{a}) \cup \{C_k\}$ for $k \in \{1, 2\}$.
$\exists$-**rule**	if:	$\exists R \cdot C \in L_G(\mathsf{a}_1)$, for some a_1 , and there is no $\mathsf{a}_2 \in V_G$, such that both $R \in L_G(\langle \mathsf{a}_1, \mathsf{a}_2 \rangle)$ and $C \in L_G(\mathsf{a}_2)$.
	then:	$\mathcal{S}' = \mathcal{S} \cup \{G'\} \setminus \{G\}$, where $G' = \langle V_G \cup \{\mathsf{a}_2\}, E_G \cup \{\langle \mathsf{a}_1, \mathsf{a}_2 \rangle\}, L_{G'} \rangle$.
		$L_{G'} = L_G$ except $L_{G'}(\mathsf{a}_2) = \{C\}$, $L_{G'}(\langle \mathsf{a}_1, \mathsf{a}_2 \rangle) = \{R\}$.
$\forall$-**rule**	if:	$\forall R \cdot C \in L_G(\mathsf{a}_1)$, for some a_1 and there is $\mathsf{a}_2 \in V_G$, such that $R \in L_G(\langle \mathsf{a}_1, \mathsf{a}_2 \rangle)$, but not $C \in L_G(\mathsf{a}_2)$.
	then:	$\mathcal{S}' = \mathcal{S} \cup \{G'\} \setminus \{G\}$, where $G' = \langle V_G, E_G, L_{G'} \rangle$.
		$L_{G'} = L_G$ except $L_{G'}(\mathsf{a}_2) = L_{G'}(\mathsf{a}_2) \cup \{C\}$.

Table 1: Completion rules used for expanding a set of $\mathcal{ALC}$ completion graphs (and not considering TBox). $G = \langle V_G, E_G, L_G \rangle$ is the completion graph chosen in the current iteration.

The algorithm does not prescribe the order, in which the rules are selected. Of course, this can significantly influence performance. E.g. non-deterministic rules ($\sqcup$-rule in case of $\mathcal{ALC}$) *should* be performed only when no other rule is applicable, to prevent generating additional completion graphs first, all of which need to be tested in CONSISTENCY/MODEL TEST steps of the algorithm. For details on other tableau optimization technique, see e.g. Chapter 9 in [1].

Correctness and Completeness I will not repeat the full proof of correctness and completeness of the algorithm, that was already presented in [1] and [3]. I only sketch main rationale behind the algorithm that helps to better understand its idea. Correctness is a direct consequence of the semantics of completion rules. E.g. if there was a model $\mathcal{I}$ corresponding to G and $\mathsf{A}_1 \sqcap \mathsf{A}_2 \in L_G(\mathsf{a})$ for some a, then, following the

semantics of $\mathcal{ALC}$, $\mathsf{a}^{\mathcal{I}} \in (\mathsf{A}_1 \sqcap \mathsf{A}_2)^{\mathcal{I}}$ and $\mathsf{a}^{\mathcal{I}} \in {\mathsf{A}_1}^{\mathcal{I}} \cap {\mathsf{A}_2}^{\mathcal{I}}$. This is ensured by putting both A_1 and A_2 into the $L'_G(\mathsf{a})$ by the $\sqcap-rule$. For the other direction and other rules the idea is similar.

Completeness is shown by constructing a canonical model $\mathcal{I}$ of $\mathcal{O}$ for each complete completion graph that does not contain a direct clash, as follows:

- The interpretation domain $\Delta^{\mathcal{I}}$ contains all graph vertices,
- for each named concept A we define $\mathsf{A}^{\mathcal{I}} = \{\mathsf{a} \mid \mathsf{A} \in L_G(\mathsf{a})\}$,
- for each named role R we define $\mathsf{R}^{\mathcal{I}} = \{\langle \mathsf{a}_1, \mathsf{a}_2 \rangle \mid \mathsf{R} \in L_G(\langle \mathsf{a}_1, \mathsf{a}_2 \rangle)\}$,

Induction according to the axiom types and complex concept structure shows that it is indeed a model of the original ontology. E.g. if $C(\mathsf{a}) \in \mathcal{O}$, then $\mathsf{a}^{\mathcal{I}} \in C^{\mathcal{I}}$ must hold for any complete graph G: (i) if $C = \mathsf{A}$ is a named concept, then indeed $\mathsf{a}^{\mathcal{I}} \in \mathsf{A}^{\mathcal{I}}$ because $\mathsf{A} \in L_{G_0}(\mathsf{a})$ (this is, how G_0 was constructed in the INITIALIZATION step of the algorithm) and $L_{G_0}(\mathsf{a}) \cup L_G(\mathsf{a})$, (ii) if $C = \mathsf{A}_1 \sqcap \mathsf{A}_2$ where both A_1 and A_2 are named concepts, then $\mathsf{a}^{\mathcal{I}} \in \mathsf{A}_1 \sqcap {\mathsf{A}_2}^{\mathcal{I}}$ and thus $\mathsf{a}^{\mathcal{I}} \in {\mathsf{A}_1}^{\mathcal{I}}$ and $\mathsf{a}^{\mathcal{I}} \in {\mathsf{A}_2}^{\mathcal{I}}$ because $\{\mathsf{A}_1, \mathsf{A}_2\} \subseteq L_{G'}$ where G' is a completion graph that resulted from application the $\sqcap$-rule. If this rule was not applied, then $G \supseteq G'$ is not complete, which contradicts our assumption. For the other axiom types and concept constructs the idea is similar.

In case of $\mathcal{ALC}$, there is a correspondence between a model and a complete completion graph simple, as presented. However, tableaux algorithms for expressive description logics require more complex structures and more complex transformations to achieve such correspondence, see e.g. [1] or [2] for more details.

Example 1 *Let's check consistency of an $\mathcal{ALC}$ ontology $\mathcal{O} = \{\alpha\}$, where α is $C(\mathsf{PillarScour})$ and C is*

$$(\exists \mathsf{isFailureOf} \cdot \mathsf{Column} \sqcap \exists \mathsf{isFailureOf} \cdot \mathsf{Pillar} \sqcap \neg \exists \mathsf{isFailureOf} \cdot (\mathsf{Pillar} \sqcap \mathsf{Column}))$$

This axiom says that PillarScour *is failure of some bridge and it is a failure of some pillar but it is not a failure of an object that is a bridge and a pillar at the same time.*

The first step is to transform the complex concept into Negational Normal Form. This produces α_2 that is $C_2(\mathsf{PillarScour})$ where C_2 is

$$(\exists \mathsf{isFailureOf} \cdot \mathsf{Column} \sqcap \exists \mathsf{isFailureOf} \cdot \mathsf{Pillar} \sqcap \forall \mathsf{isFailureOf} \cdot (\neg \mathsf{Pillar} \sqcap \neg \mathsf{Column}))$$

α_2 is semantically equivalent with α (i.e. $\{\alpha\} \models \{\alpha_2\}$ and $\{\alpha_2\} \models \{\alpha\}$), but it has negations only before named concepts.

The initial state of the algorithm is $\mathcal{S}_0 = \{G_0\}$, where graph

$$G_0 = \langle \{\mathsf{PillarScour}\}, \emptyset, \{\mathsf{PillarScour} \mapsto \{C_2\}\} \rangle \tag{1}$$

is shown in Figure 1.

At this point, the algorithm passes four sequences of the steps CONSISTENCY TEST → MODEL TEST → RULE APPLICATION of the tableau algorithm, that

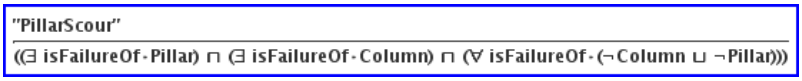


Figure 1: Initial state of the tableau algorithm.

can be denoted as a state evolution during the step RULE APPLICATION (the label over the arrow denotes the rule that was used):

$$\{G_0\} \xrightarrow{\sqcap\text{-}rule} \{G_1\} \xrightarrow{\exists\text{-}rule} \{G_2\} \xrightarrow{\exists\text{-}rule} \{G_3\} \xrightarrow{\forall\text{-}rule} \{G_4\},$$

where G_4 is depicted in Figure 2.

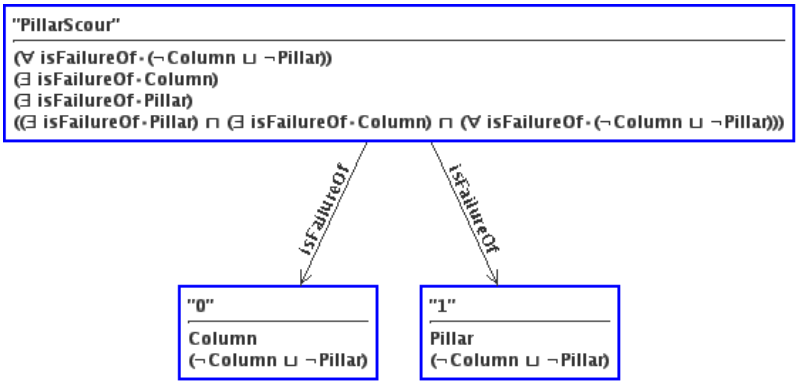
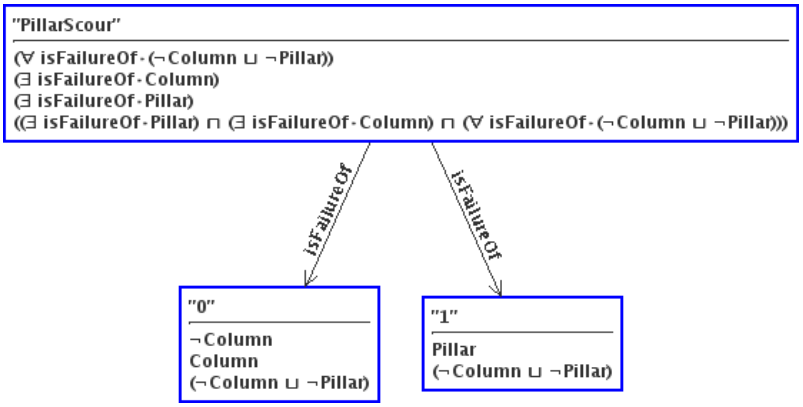


Figure 2: Graph G_4 evolved deterministically from G_0 using $\sqcap$-rule, $\exists$-rule, $\forall$-rule.

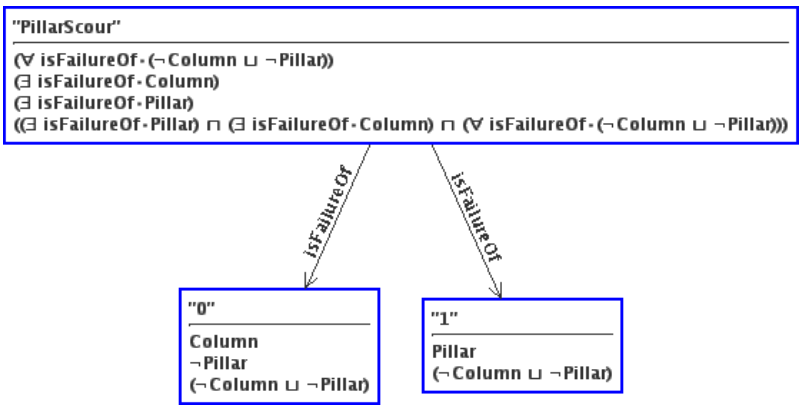
So far, only deterministic rules (i.e. those that do not increase the number of completion graphs) have been used. Looking carefully at Figure 2, it is clear that the only rule that remains applicable is the $\sqcup$-rule. The rule can be applied on the concept $(\neg Column \sqcup \neg Pillar)$ in the label of either vertex 0 *or* 1*. Picking e.g.* 0 *and applying $\sqcup$-rule produces a new state $\{G_5, G_6\}$ depicted in Figure 3.*

Graph G_5 contains a direct clash, as Column *and* ¬Column *is in the label of vertex* 0*, and thus G_5 is discarded, as it cannot be transformed to a model (e.g. the canonical model defined in relation to the completeness of the algorithm). Thus, G_6 is picked and $\sqcup$-rule is applied, which results in a new state $\{G_7, G_8\}$, as shown in Figure 4.*

While G_7 contains a direct clash in vertex 1*, completion graph G_8 is complete with respect to the $\mathcal{ALC}$ completion rules and does not contain a direct clash. Thus, a canonical model $\mathcal{I}_1 = \langle \Delta^{\mathcal{I}_1}, \cdot^{\mathcal{I}_1} \rangle$ can be constructed from G_8 as follows:*



(a) Graph G_5

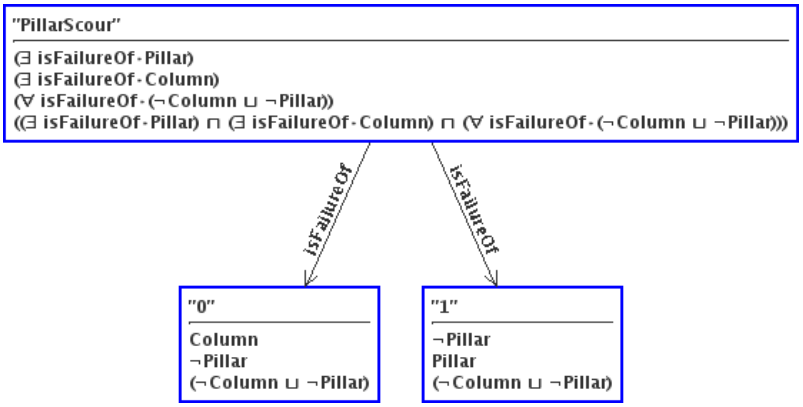


(b) Graph G_6

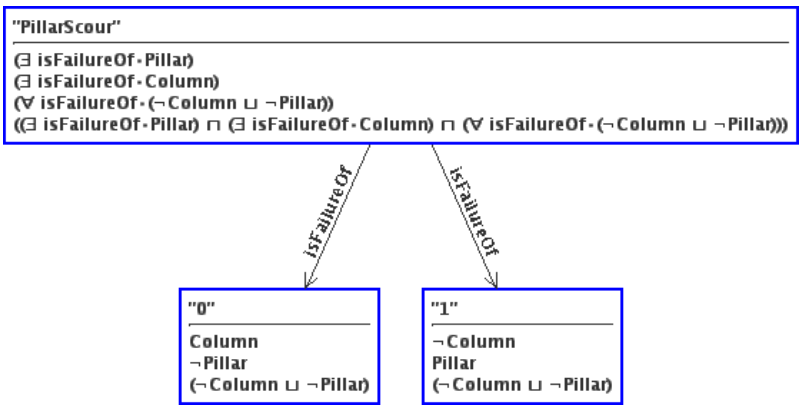
Figure 3: Graphs G_5 and G_6 produced by application of the $\sqcup$-rule on vertex 0.

$$
\begin{aligned}
\Delta^{\mathcal{I}_1} &= \{\mathsf{PillarScour}, 0, 1\} \\
\mathsf{isFailureOf}^{\mathcal{I}_1} &= \{\langle \mathsf{PillarScour}, 0\rangle, \langle \mathsf{PillarScour}, 1\rangle\} \\
\mathsf{Pillar}^{\mathcal{I}_1} &= \{1\} \\
\mathsf{Column}^{\mathcal{I}_1} &= \{0\} \\
\mathsf{PillarScour}^{\mathcal{I}_1} &= \{\mathsf{PillarScour}\}
\end{aligned}
$$

$\mathcal{I}_1$ is not the only intepretation of $\mathcal{O}$ - another model of $\mathcal{O}$ might be $\mathcal{I}_2$, for which $\mathsf{Column} = \{0, \mathsf{PillarScour}\}$ *and coincides with $\mathcal{I}_1$ on the rest. This documents the fact that a complete completion graph corresponds to one (canonical) model, but might*



(a) Graph G_7



(b) Graph G_8

Figure 4: Graphs G_7 and G_8 produced by application of the $\sqcup$-rule on vertex 1.

correspond to other models as well.

2 Tableau algorithm for general $\mathcal{ALC}$

In the general case, the situation is slightly complicated. The TBox knowledge is included in the algorithm by an extra rule

$\sqsubseteq$-**rule**	if:	$(C_1 \sqsubseteq C_2) \in \mathcal{T}$ and $(\neg C_1 \sqcup C_2) \notin L_G(\mathsf{a})$ for some a that is not blocked.
	then:	$\mathcal{S}' = \mathcal{S} \cup \{G'\} \setminus \{G\}$, where $G' = \langle V_G, E_G, L_{G'} \rangle$.
		$L_{G'} = L_G$ except $L_{G'}(\mathsf{a}) = L_G(\mathsf{a}) \cup \{\neg C_1 \sqcup C_2\}$.

For this rule to be applicable, each equivalence axiom $C_1 \equiv C_2 \in \mathcal{T}$ has to be replaced by two subsumption axioms $C_1 \sqsubseteq C_2$ and $C_2 \sqsubseteq C_1$ in the PREPROCESSING step of the algorithm. Unfortunately, such simple extension of the tableau algorithm need not terminate. E.g. consider ontology $\mathcal{O} = \{\mathsf{Object} \sqsubseteq \exists \mathsf{hasPart} \cdot \mathsf{Object}, \mathsf{Object}(\mathsf{CharlesBridge})\}$.

2.1 Blocking

To ensure termination of the algorithm it is necessary to detect cycles of the generated model that might occur due to the application of the $\sqsubseteq$-rule. The cycles are detected using *blocking* that ensures that a completion graph, although representing possibly infinite model, is always finite. This is ensured by preventing the inference rules to generate node and edge patterns that "repeat regularly". The notion of regularity is different for each description logic; for $\mathcal{ALC}$,so called *subset blocking* [1] is used:

> A vertex a_1 in a completion graph G, but not occuring in $\mathcal{A}$, *is blocked* by a vertex a_2, if there is an oriented path in G from a_2 to a_1 and a $L_G(\mathsf{a}_1) \subseteq L_G(\mathsf{a}_1)$.

Then all rules are applicable *only* on an individual, only if this individual is not blocked. As a result we get a set of completion rules in $\mathcal{ALC}$ with general TBox, as shown in Table 2

$\sqsubseteq$-**rule**	if:	$(C_1 \sqsubseteq C_2) \in \mathcal{T}$ and $(\neg C_1 \sqcup C_2) \notin L_G(\mathsf{a})$ for some a **that is not blocked**.
	then:	$\mathcal{S}' = \mathcal{S} \cup \{G'\} \setminus \{G\}$, where $G' = \langle V_G, E_G, L_{G'}\rangle$. $L_{G'} = L_G$ except $L_{G'}(\mathsf{a}) = L_G(\mathsf{a}) \cup \{\neg C_1 \sqcup C_2\}$.
$\sqcap$-**rule**	if:	$(C_1 \sqcap C_2) \in L_G(\mathsf{a})$, for some a **that is not blocked** and $\{C_1, C_2\} \nsubseteq L_G(\mathsf{a})$.
	then:	$\mathcal{S}' = \mathcal{S} \cup \{G'\} \setminus \{G\}$, where $G' = \langle V_G, E_G, L_{G'}\rangle$. $L_{G'} = L_G$ except $L_{G'}(\mathsf{a}) = L_G(\mathsf{a}) \cup \{C_1, C_2\}$.
$\sqcup$-**rule**	if:	$(C_1 \sqcup C_2) \in L_G(\mathsf{a})$, for some a **that is not blocked** and $\{C_1, C_2\} \cap L_G(\mathsf{a}) = \emptyset$.
	then:	$\mathcal{S}' = \mathcal{S} \cup \{G_1, G_2\} \setminus \{G\}$, where $G_k = \langle V_G, E_G, L_{G_k}\rangle$. $L_{G_k} = L_G$ except $L_{G_k}(\mathsf{a}) = L_G(\mathsf{a}) \cup \{C_k\}$ for $k \in \{1, 2\}$.
$\exists$-**rule**	if:	$\exists R \cdot C \in L_G(\mathsf{a}_1)$, for some a_1 **that is not blocked**, and there is no $\mathsf{a}_2 \in V_G$, such that both $R \in L_G(\langle \mathsf{a}_1, \mathsf{a}_2\rangle)$ and $C \in L_G(\mathsf{a}_2)$.
	then:	$\mathcal{S}' = \mathcal{S} \cup \{G'\} \setminus \{G\}$, where $G' = \langle V_G \cup \{\mathsf{a}_2\}, E_G \cup \{\langle \mathsf{a}_1, \mathsf{a}_2\rangle\}, L_{G'}\rangle$. $L_{G'} = L_G$ except $L_{G'}(\mathsf{a}_2) = \{C\}$, $L_{G'}(\langle \mathsf{a}_1, \mathsf{a}_2\rangle) = \{R\}$.
$\forall$-**rule**	if:	$\forall R \cdot C \in L_G(\mathsf{a}_1)$, for some a_1 **that is not blocked** and there is $\mathsf{a}_2 \in V_G$, such that $R \in L_G(\langle \mathsf{a}_1, \mathsf{a}_2\rangle)$, but not $C \in L_G(\mathsf{a}_2)$.
	then:	$\mathcal{S}' = \mathcal{S} \cup \{G'\} \setminus \{G\}$, where $G' = \langle V_G, E_G, L_{G'}\rangle$. $L_{G'} = L_G$ except $L_{G'}(\mathsf{a}_2) = L_{G'}(\mathsf{a}_2) \cup \{C\}$.

Table 2: Completion rules used for expanding a set of $\mathcal{ALC}$ completion graphs. $G = \langle V_G, E_G, L_G\rangle$ is the completion graph chosen in the current iteration.

References

[1] Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter Patel-Schneider, editors. *The Description Logic Handbook, Theory, Implementation and Applications*. Cambridge, 2003.

[2] Ian Horrocks, Oliver Kutz, and Ulrike Sattler. The Even More Irresistible SROIQ. In Patrick Doherty, John Mylopoulos, and Christopher A. Welty, editors, *KR*, pages 57–67. AAAI Press, 2006.

[3] Manfred Schmidt-Schauss and Gert Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48(1):1–26, February 1991.