1 Preliminaries

Let $\mathbf{C}$ be a set of concept names and $\mathbf{R}$ a set of role names such that they are disjoint.

Definition 1 ($QFBAPA$)**.** Let T be a set of symbols

- set terms over T are:
  - empty set $\emptyset$ and universal set $\mathcal{U}$
  - every set symbol in T
  - if s, t are set terms then also $s \cap t$, $s \cup t$ and $s^{\neg}$
- set constraints over T are
  - $s \subseteq t$ and $s \not\subseteq t$
  - $s = t$ and $s \neq t$

  where s, t are set terms
- cardinality terms over T are:
  - every number $n \in \mathbb{N}$
  - $|s|$ if s is a set term
  - if k, l are cardinality terms then also $k + l$ and $n \cdot k$, $n \in \mathbb{N}$
- cardinality constraints over T are:
  - $k = l$ and $k \neq l$
  - $k < l$ and $k \geq l$
  - $k \leq l$ and $k > l$
  - $n\ dvd\ k$ and $n\ \neg dvd\ k$

  where k, l are cardinality terms and $n \in \mathbb{N}$

For readability we use $\lesseqgtr$ to address the comparison symbols $=$, $\leq$, $\geq$, $<$, $>$. The negation $\not\lesseqgtr$ address the symbols $\neq$, $>$, $<$, $\geq$, $\leq$ respectively.

Definition 2 ($\mathcal{ALCSCC}$)**.** Concepts are:

- all concept names
- $succ(c)$ if c is a set or cardinality constraint over $\mathcal{ALCSCC}$ concepts and role names
- if C, D are concepts then:
  - $\neg C$
  - $C \sqcup D$
  - $C \sqcap D$

Definition 3 (Negation Normal Form)**.** A concept is in *negation normal form* (NNF) if the negation sign $\neg$ appears only in front of a concept name or above a role name. Let C be a arbitrary concept. Its NNF is obtained by applying the following rules

- $\neg\top \rightarrow \bot$
- $\neg\bot \rightarrow \top$
- $\neg\neg C \rightarrow C$
- $\neg(C \sqcap D) \rightarrow \neg C \sqcup \neg D$
- $\neg(C \sqcup D) \rightarrow \neg C \sqcap \neg D$
- $(C)^{\neg} \rightarrow \neg C$
- $\neg succ(c) \rightarrow succ(\neg c)$
- $\neg(k \lesseqgtr l) \rightarrow k \not\lesseqgtr l$
- $\neg(n\ dvd\ k) \rightarrow n\ \neg dvd\ k$
- $\neg(n\ \neg dvd\ k) \rightarrow n\ dvd\ k$
- $\neg(s_1 \subseteq s_2) \rightarrow |s_1 \cap s_2^{\neg}| \geq 1$
- $(s \cap t)^{\neg} \rightarrow s^{\neg} \cup t^{\neg}$
- $(s \cup t)^{\neg} \rightarrow s^{\neg} \cap t^{\neg}$
- $(s^{\neg})^{\neg} \rightarrow s$

With $NNF(C)$ we denote the concept which is obtained by applying the rules above on C until none is applicable any more.

Definition 4 (Positive and Negative Sign)**.** Let be $(x, y) : s$ an assertion. A concept name C or a role name r has a *positive sign* in s if it occurs with no negation sign in front or above it in s. It has a *negative sign* otherwise. A concept name C (or a role name r) can have both sign in s if C and $\neg C$ (or r and $r^{\neg}$) are in s.

The constraint set S is a finite set of assertions of the form $x : C$ and $(x, y) : s$, where C is a concept, s a set term and x, y variables. The set $Var(S)$ is the set of variables occurring in S.
Note that in a assertion in NNF the negation sign can only occurs in front of a concept name and above a role name.

Definition 5 (Interpretation)**.** An *interpretation* $\mathcal{I} = (\cdot^{\mathcal{I}}, \cdot^{\mathcal{I}}, \pi_{\mathcal{I}})$ over a constraint set S in $\mathcal{ALCSCC}$ consists of a non-empty set $\Delta^{\mathcal{I}}$, an assignment $\pi_{\mathcal{I}}$ and a mapping $\cdot^{\mathcal{I}}$ which maps:

- $\emptyset$ to $\emptyset^{\mathcal{I}}$
- $\mathcal{U}$ to $\mathcal{U}^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
- each variable $x \in Var(S)$ to $x^{\mathcal{I}} \in \Delta^{\mathcal{I}}$
- every concept names $A \in \mathbf{C}$ to $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
- every role name $r \in \mathbf{R}$ to $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, such that every element in $\Delta^{\mathcal{I}}$ has a finite number of successors.

The set $r^{\mathcal{I}}(x)$ contains all elements y such that $(x, y) \in r^{\mathcal{I}}$ e.g. it contains all r-successors of x.
For compound concepts the mapping $\cdot^{\mathcal{I}}$ is extended inductively as follows

- $\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$ and $\bot^{\mathcal{I}} = \emptyset^{\mathcal{I}}$
- $(C \sqcap D)^{\mathcal{I}} := C^{\mathcal{I}} \cap D^{\mathcal{I}}$, $(C \sqcup D)^{\mathcal{I}} := C^{\mathcal{I}} \cup D^{\mathcal{I}}$
- $(\neg C)^{\mathcal{I}} := \Delta^{\mathcal{I}} \backslash C^{\mathcal{I}}$
- $(s \cap t)^{\mathcal{I}} := s^{\mathcal{I}} \cap t^{\mathcal{I}}$, $(s \cup t)^{\mathcal{I}} := s^{\mathcal{I}} \cup t^{\mathcal{I}}$
- $(s^{\neg})^{\mathcal{I}} := \mathcal{U}^{\mathcal{I}} \backslash s^{\mathcal{I}}$
- $|s|^{\mathcal{I}} := |s^{\mathcal{I}}|$
- $(k + l)^{\mathcal{I}} := (k^{\mathcal{I}} + l^{\mathcal{I}})$, $(n \cdot k)^{\mathcal{I}} := n \cdot k^{\mathcal{I}}$
- $succ(c)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} | \text{the mapping } \cdot^{\mathcal{I}_x} \text{ satisfies } c\}$

The mapping $\cdot^{\mathcal{I}_x}$ maps $\emptyset$ to $\emptyset^{\mathcal{I}}$, $\mathcal{U}$ to $\mathcal{U}^{\mathcal{I}_x} := \{\bigcup_{r \in \mathbf{R}} r^{\mathcal{I}}(x)\}$, every concept C occurring in c to $C^{\mathcal{I}_x} := C^{\mathcal{I}} \cap \mathcal{U}^{\mathcal{I}_x}$ and every role name r occurring in c to $r^{\mathcal{I}_x} := r^{\mathcal{I}}(x)$.
The mappings satisfies for the set terms s, t and the cardinality terms k, l

- $s = t$ iff $s^{\mathcal{I}} = t^{\mathcal{I}}$
- $s \subseteq t$ iff $s^{\mathcal{I}} \subseteq t^{\mathcal{I}}$
- $k \lesseqgtr l$ iff $k^{\mathcal{I}} \lesseqgtr l^{\mathcal{I}}$
- $n \, dvd \, l$ iff $\exists m \in \mathbb{N} : n \cdot m = l^{\mathcal{I}}$

The *assignment* $\pi_{\mathcal{I}} : Var(S) \to \Delta^{\mathcal{I}}$ satisfies

- $x : C$ iff $\pi_{\mathcal{I}}(x) \in C^{\mathcal{I}}$
- $(x, y) : s$ iff $(\pi_{\mathcal{I}}(x), \pi_{\mathcal{I}}(y)) \in s^{\mathcal{I}}$

$\pi_{\mathcal{I}}$ satisfies a constraint set S if $\pi_{\mathcal{I}}$ satisfies every assertion in S. If $\pi_{\mathcal{I}}$ satisfies S then $\mathcal{I}$ is a model of S.

By the semantic definition $s^{\mathcal{I}} \cup (s^{\neg})^{\mathcal{I}} = \mathcal{U}^{\mathcal{I}}$.

Definition 6 (Number of successors)**.** Let S be a set of assertion, $\mathcal{I}$ be an interpretation, x be a variable and k be a cardinality term. The number of successors of x in k in the interpretation $\mathcal{I}$ is denoted by $n_{\mathcal{I}}(x, k, S) := k^{\mathcal{I}_x}$ where i, j, l are cardinality terms, $n \in \mathbb{N}$ and s is a set term.
A assertion regarding a variable x in an interpretation $\mathcal{I}$ is *violated* if

- $x : succ(k \lesseqgtr n)$ and $n_{\mathcal{I}}(x, k, S) \not\lesseqgtr n$
- $x : succ(k \lesseqgtr l)$ and $n_{\mathcal{I}}(x, k, S) \not\lesseqgtr n_{\mathcal{I}}(x, l, S)$
- $x : succ(n \, dvd \, k)$ and $mod(n_{\mathcal{I}}(x, k, S), n) \neq 0$

where $n \in \mathbb{N}$.

2 Tableau

For the algorithm we assume that the assertions in the constraint set are in NNF. Similar in [1] and [2], where a variable can be replaced by another variable, we can merge two variables during the Tableau-algorithm.

Definition 7 (Merge)**.** *Merging* y_1 *and* y_2 *results in one variable* y*: replace all occurrence of* y_1 *and* y_2 *with* y*.*

Note that by merging two successors other assertions might become violated:

$$\begin{aligned} S = \{&x : succ(|r \cap A| = 1) \sqcap succ(|r \cap B| = 1) \sqcap succ(|r| > 1),\\ &y_1 : A, y_2 : B, x.r.y_1, x.r.y_2\} \end{aligned} \tag{1}$$

If we merge y_1 and y_2 then the assertion $x : succ(|r| > 1)$ which was satisfied becomes violated.
But not only assertions regarding x might become violated after merging two successors of x:

$$\begin{aligned} S = \{&x : succ(|r| \leq 1), x.r.y_1, x.r.y_2,\\ &y_1 : succ(|s| \leq 1), y_2 : succ(|s| \leq 1), y_1.s.z_1, y_2.s.z_2\} \end{aligned} \tag{2}$$

We see that the first assertion is violated and therefore merging y_1 and y_2 to y would solve the problem but on the other hand the assertion regarding y become violated:

$$S = \{x : succ(|r| \leq 1), x.r.y, y : succ(|s| \leq 1), y.s.z_1, y.s.z_2\}$$

To solve this problem we have to merge z_1 and z_2.

Definition 8 (Induced Interpretation $\mathcal{I}(S)$)**.** *An interpretation* $\mathcal{I}(S)$ *can be induced from a constraint set* S *by the following steps:*

- *for each variable* $x \in Var(S)$ *we introduce* $x^{\mathcal{I}(S)}$ *and add it to* $\Delta^{\mathcal{I}(S)}$
- *for each* $x : C$ *such that* C *is a concept name we add* $x^{\mathcal{I}(S)}$ *to* $C^{\mathcal{I}(S)}$
- *for each* $(x, y) : r$ *such that* r *is a role name we add* $(x^{\mathcal{I}(S)}, y^{\mathcal{I}(S)})$ *to* $r^{\mathcal{I}(S)}$

With $\mathcal{I}(S)$ we can count how many successors a variable has during the Tableau-algorithm. Regarding the cardinality constraints $k \leq l$ and $k < l$ we have to be *safe* before adding or merging variables. Let S' be the obtained constraint set from S:

- It is safe to introduce a new variable y if

$$n_{\mathcal{I}(S)}(x, k, S) - n_{\mathcal{I}(S)}(x, l, S) \neq n_{\mathcal{I}(S')}(x, k, S') - n_{\mathcal{I}(S')}(x, l, S')$$

- It is safe to merge two variable y_1, y_2 if by merging them we do not violate any assertions regarding x, for which we have $(x, y_1) : s \in S$ and $(x, y_2) : s \in S$, in $\mathcal{I}(S)$

For the next definition we define first properties of the following notations:

- Conjunction binds stronger than disjunction: $s \cup t \cap u = s \cup (t \cap u)$
- if k, l are cardinality terms then $k = l$ replaces $k \leq l$ and $k \geq l$
- if s, t are set terms then $s = t$ replaces $s \subseteq t$ and $s \supseteq t$

To maintain readability we write $k \leq l$ instead of $l \geq k$ and $k < l$ instead of $l > k$.

Definition 9 (Tableau)**.** Let S be a set of assertions in NNF.

1. $\sqcap$-rule: S contains $x : C_1 \sqcap C_2$ but not both $x : C_1$ and $x : C_2$
   $\rightarrow S := S \cup \{x : C_1, x : C_2\}$
2. $\sqcup$-rule: S contains $x : C_1 \sqcup C_2$ but neither $x : C_1$ nor $x : C_2$
   $\rightarrow S := S \cup \{x : C_1\}$ or $S := S \cup \{x : C_2\}$
3. *choose*-rule: S contains $x : succ(k \leq l)$ and $(x, y) : k'$ with k' in k but $(x, y) : s \notin S$ for some $|s|$ in k
   $\rightarrow$ for all $|s|$ in k, in which k' occurs, either $S := S \cup \{(x, y) : s\}$ or $S := S \cup \{(x, y) : s^{\neg}\}$ and then jump to rule 7
4. *choose-a-role*-rule: S contains $(x, y) : s$ but there is no $(x, y) : r \notin S$, $r \in \mathbf{R}$, where r has a positive sign in this assertion
   $\rightarrow$ choose one rule name $r \in \mathbf{R}$, which does not has a negative sign in s, and add $(x, y) : r$ to S
5. *cardinality*-rule: S contains $x : succ(c)$, with $c \in \{k \leq l,\ k < l,\ n\,dvd\,l\}$, such that c is violated in $\mathcal{I}(S)$ regarding x
   a) if there is a set term $|s|$ in l such that it is safe to add variables regarding c
      $\rightarrow$ introduce new variable y and $S := S \cup \{(x, y) : s\}$, then jump to rule 7
   b) if $l \in \mathbb{N}$ does not contain a set term and we have two successor $y_1 \neq y_2$ of x such that for a $|s|$ in k we have $(x, y_1) : s \in S$ and $(x, y_2) : s \in S$ and it is safe to merge them:
      $\rightarrow$ merge y_1 and y_2
6. *set*-rule: S contains $x : succ(s_1 \subseteq s_2)$ and $(x, y) : s_1$ but not $(x, y) : s_2$
   $\rightarrow S := S \cup \{(x, y) : s_2\}$ and then jump to rule 7
7. *set.term*-rule (Repeat until inapplicable): In S is $(x, y) : s$ and
   a) $s = s_1 \cap s_2$ but $\{(x, y) : s_1,\ (x, y) : s_2\} \nsubseteq S$
      $\rightarrow S := S \cup \{(x, y) : s_1,\ (x, y) : s_2\}$
   b) $s = s_1 \cup s_2$ and neither $\{(x, y) : s_1\} \subseteq S$ nor $S\{(x, y) : s_2\} \subset S$
      $\rightarrow$ either $S := S \cup \{(x, y) : s_1\}$ or $S := S \cup \{(x, y) : s_2\}$
   c) $s = C$ and $y : C \notin S$, where C is an $\mathcal{ALCSCC}$ concepts
      $\rightarrow S := S \cup \{y : C\}$

Note that:

- s in 5a can also be of the form $t^\neg$.
- 5b is never applicable for $n\, dvd\, l$
- if $n_1\, dvd\, n_2 \cdot l$ and $mod(n_2, n_1) \neq 0$ then $n_1\, dvd\, l$ eventually

Definition 10 (Clash)**.** A constraint set S contains a *clash* if

- $\{x : \perp\} \subseteq S$ or
- $\{x : A, x : \neg A\} \subseteq S$ or
- $\{x : succ(c)\} \subseteq S$ and c is violated regarding x

and no more rules are applicable.

Definition 11 (Derived Set)**.** A *derived set* is a constraint set S' where rule 7 is not applicable.

The first rule decompose the conjunction and the second rule adds non-deterministically the right assertion. The third rule is important because we need to know of every successor what kind of role successors they are and in which concepts they are. We use $n_{\mathcal{I}}(x, k, S)$ to count the successors of x in k which is important for detecting and avoiding violations of assertions. Now there might be a successor y which satisfies only some part of k in the given S such that $n_{\mathcal{I}}(x, k, S)$ does not count y. However there might be an interpretation $\mathcal{I}'$ such that $n_{\mathcal{I}'}(x, k, S)$ counts y and hence $n_{\mathcal{I}}(x, k, S) \neq n_{\mathcal{I}'}(x, k, S)$. However the Tableau-algorithm should be able to construct every interpretation of S. Therefore this rule adds non-deterministically either $(x, y) : s$ or $(x, y) : s^\neg$ which are the only two possibilities.
The *choose-a-role*-rule is necessary because for a assertion $x : succ(c)$ we might have no role name with a positive sign in c. Which means we know x must have some successors but we can not decide which role-successor it is. As example we have

Example 1.

$$\mathbf{R} = \{r, s\}$$
$$S = \{x : succ(|r^\neg| \geq 1)\}$$

It states that x have at least one successor which is not a r-successor. Since $\mathbf{R}$ only contains r and s we know that the successors must be s-successors. With that rule we would pick either r or s. We can not pick r because $r^\neg$ occurs in the assertion. Therefore we have to pick s.

We consider now two examples to explain the rule. 5b

Example 2. Consider the following example

$$S = \{x : succ(|r| = 1) \sqcap succ(|r \cap s| = 1) \sqcap succ = (|r \cap C| = 1|)$$
$$x : succ(|r| = 1), x : succ(|r \cap s| = 1), x : succ = (|r \cap C| = 1|)\}$$

If we try to satisfy all the assertions in the second line with rule 5a we end up with at least one assertion being violated: The problem is that by applying rule 5a we add one successor y which always raise $|r|$. If we apply 5a one time nothing the amount of violated assertions decreases. If we apply it a second time then $x : succ(|r| = 1)$ is violated. In case we decided to apply 5a on $x : succ(|r| = 1)$ and $x : succ(|r \cap s| = 1)$ we have the option of either applying rule 5a on $x : succ(|r \cap C| = 1)$ or applying rule 5b on the two new variables. We decide for the first choice:

$$\begin{tikzcd}
 & x \arrow[dl, "r"'] \arrow[d, "r \cap s"] \arrow[dr, "r"] & \\
y_1 & y_2 & y_3 \\
 & & C
\end{tikzcd}$$

In this case we have to merge three variables: y_1, y_2 and y_3. We can apply 5b here because satisfied assertions regarding x does not become violated. We can also see that the order does not matter: In case we decided for the second choice and merge y_1 and y_2 to y before introducing y_3 we end up with the same results because then we have to merge y and y_3.

$$\begin{tikzcd}
x \arrow[r, "r \cap s"] & y \\
 & C
\end{tikzcd}$$

In the example we see a case where S is unsatisfiable.

Example 3.

$$S = \{x : succ(|r| < 2) \sqcap succ(|r| \geq 2)\}$$

First we apply the rules 5a two times to add two r-successors for x to satisfy $x : succ(|r| \geq 2)$. But this results to $x : succ(|r| < 2)$ being violated. If we ignore the condition in rule 5b and apply it then we merge the two successors leading to $x : succ(|r| < 2)$ being satisfied but $x : succ(|r| \geq 2)$ being violated. Then we can apply the rule 5a again leading to $x : succ(|r| < 2)$ being violated and so on. By the condition in 5b we can not merge the two new variables because the satisfied assertion $x : succ(|r| \geq 2)$ would become violated. Hence the algorithm terminates with a clash stating that the constraint set is unsatisfiable.

We also know that the application of rule 7 eventually terminates because the number of concept names and role names are finite in S (since S is finite).

3 Correctness

For the correctness proof of the Tableau-algorithm we have to show that

- for every input the Tableau-algorithm terminates
- If no more rules are applicable on a clash-free constraint set S then S is satisfiable
- For every interpretation $\mathcal{I}$ of a satisfiable constraint set S we can find a chain of rule application such that the induced interpretation $\mathcal{I}(S)$ is equal to $\mathcal{I}$

First we prove that the tableau algorithm terminates.

Proposition 1. Let C be a concept in negation normal form. Then there is no infinite chain of applications of any tableau rules issuing from $\{x : C\}$.

To prove this we map any derived set S to an element $\Psi(S)$ from a set Q. We then show that the elements in Q can be ordered by a well-founded relation $\prec$. A well-founded relation says that there is no infinite decreasing chain. If we can show that by obtaining a derived set S' from another set S we have $\Psi(S') \prec \Psi(S)$ then the algorithm terminates. The elements in Q are finite multisets of septuples and the elements of the septuples are either integers or mutlisets of integers. For two septuples $q = (q_1, \dots, q_7)$ and $q' = (q'_1, \dots, q'_7)$ it holds $q \prec q'$ if for the first i, $1 \leq i \leq 7$, for which q_i and q'_i differs it holds that $q_i \prec q'_i$ (also called lexicographical ordering). For two mutlisets of integers q_i and q'_i it holds $q'_i \prec q_i$ if q'_i can be obtained from q_i by replacing an integer c in q'_i by a finite number of integers which are all smaller than c. The relation $\prec$ for those multisets is also well-founded because we work with integers. That means from a multiset $\{0, \dots, 0\}$ we can not obtain a smaller multiset because we would have to replace at least one 0 with integers which are smaller.
For a concept C its size $size(C)$ is inductively defined as

- 0, if C is $\bot$
- 1, if C is a concept name of $\mathbf{C}$
- $size(\neg C) = 1 + size(C)$
- $size(succ(c)) = 1 + \sum_{C \in \mathbf{C} \text{ in c}} size(C)$
- $size(C \sqcap D) = size(C \sqcup D) = size(C) + size(D)$

The number $n_{sc}(x)$ denotes the number of assertions of the form $x : succ(s_1 \subseteq s_2)$ for a variable x. Let y be a successor of x. The number $n_{sc}(x, y)$ denotes the number of set assertions of the form $x : succ(c_1 \subseteq c_2)$ where $(x, y) : s_1 \in S$ and $(x, y) : s_2 \in S$ hold.
The asymmetrical difference of two numbers n, m is denoted by

$$n \trianglelefteq m \begin{cases} n - m & \text{if } n > m \\ 0 & \text{if } n \leq m \end{cases}$$

The septuples in Q are defined as follows

Definition 12. Let S be a constraint set. The multiset $\Psi(S)$ consist of septuples $\psi_S(x)$ for each variable x. The component of the septuples are structured as follows

- the first component is a non-negative integer $max\{size(C) \mid x : C \in S\}$
- the second component is a multiset of integers containing for each $x : C \sqcap D$, on which the $\sqcap$-rule is applicable, the non-negative integer $size(C \sqcap D)$ (respectively for $C \sqcup D$)
- the third component is a multiset which denotes for every $x : succ(k \leq l)$ the integer $n_{\mathcal{I}(S)}(x, k, S) \trianglelefteq n_{\mathcal{I}(S)}(x, l, S)$
- the fourth component is a multiset of integers in which for each successor y of x we have $n_{sc}(x) - n_{sc}(x, y)$
- the fifth component denotes the number of all successors of x in S
- the sixth component is a multiset of integers containing for each $x : succ(k \leq n) \in S$ the number of all successors y of x such that we have $(x, y) : k'$, k' occurs in k but for at least one $|s|$ in k we have neither $(x, y) : s \in S$ nor $(x, y) : \neg s \in S$
- the seventh component saves the difference of the number of all successors and the number of successors y for which there exists a positive role name r such that $(x, y) : r \in S$

Lemma 1. The following properties hold

1. For any concept C we have $size(C) \geq size(NNF(\neg C))$
2. Any variable y in a derived set S has at most one predecessor x in S
3. If $(x, y) : r \in S$ for a $r \in \mathbf{R}$ (and y is a introduced variable) then

$$max\{size(C) \mid x : C \in S\} > max\{size(D) \mid y : D \in S\}$$

Proof.

1. By induction over the number of applications to compute the negation normal form we have $size(C) = size(NNF(\neg C))$. Because $\neg succ(k \geq 0)$ can be replace by $\perp$ which is *smaller* than $\neg succ(k \geq 0)$, we have $size(C) \geq size(NNF(\neg C))$. This can be done because $\neg succ(k \leq 0) = succ(k < 0)$ which is impossible to satisfy and therefore $\neg succ(k \leq 0) = \perp$.
2. If y is a newly introduced variable, then it can only be introduced by exactly one variable x which is y's only predecessor. If two variables are merged together by rule 5b then both variables must have the same predecessor x by the condition of that rule.

3. By the second fact we know that x is the only predecessor of y. When y is introduced by applying 5a on a assertion $x : succ(k \lesseqgtr l)$ then we have $y : C$ for every concept C occurring in l (for $\neg C$ we have $y : \neg C$). We know that $size(succ(k \lesseqgtr l))$ is greater then $size(C) =: max\{size(D) \mid y : D \in S\}$ therefore Lemma 1.3 holds. A new assertion $y : D$ can occur either because rule 1 or 2 are applicable on $y : C$ with $C = D \sqcap D'$ or $C = D \sqcup D'$, which neither raise $max\{size(D) \mid y : D \in S\}$, or because rule 3 is applicable but that also does not raise $max\{size(D) \mid y : D \in S\}$: If rule 3 is applicable on $x : succ(k \leq l)$ then for every added assertion $y : D$ the concept D must occur in k and therefore $size(succ(k \leq)) > size(D)$. If y gets merged together with another variable z, then y and z must have the same predecessor which means that all concept sizes regarding z are also smaller then $max\{size(C) \mid x : C \in S\}$.

□

From the next Lemma we can conclude that the Tableau-algorithm terminates.

Lemma 2. If S' is a derived set obtained from the derived set S, then $\Psi(S') \prec \Psi(S)$

The following proof is sectioned by the definition of obtaining a derived set.

Proof.

1. S' is obtained by the application of rule 1 on $x : C \sqcap D$:
   The first component remains the same because $size(C) < size(C \sqcup D)$ and $size(D) < size(C \sqcap D)$. The second component decreases because rule 1 can not be applied on $x : C \sqcap D$ any more meaning that the corresponding entry in the multiset is removed. If C (or D) happens to be a disjunction $(C' \sqcup D')$ or a conjunction $(C' \sqcap D')$ then the second component also becomes smaller because $size(C')$ and $size(D')$ are always smaller than the disjunction or conjunction of them and therefore also smaller than $size(C \sqcap D)$. Hence the entry for $size(C \sqcap D)$ can be replace by the smaller $size(C' \sqcup D')$ or $size(C' \sqcap D')$.
   Consider now a tuple $\psi_S(y)$ such that $x \neq y$. $\psi_S(y)$ can only be affected if x is a successor of y. The first and second component of $\psi_S(y)$ remain unaffected because both are independent from x. The third component can decrease but never increase: By adding an assertion for x the number $n_{sc}(y, x)$ might increases and hence the component also might decreases. The fourth, fifth and sixth component also remain unchanged because the number of y's successors does not change. The sixth also do not change because we do not add an assertion of the form $(y, x) : s$. Hence $\psi_S(y)$ does not change.
   This means that we can obtain $\Psi(S')$ from $\Psi(S)$ by replacing $\psi_S(x)$ with the smaller tuple $\psi_{S'}(x)$.

2. S' is obtained by the application of rule 2 on $x : C \sqcup D$:
   similar to above

3. S' is obtained by the application of rule 3 on $x : succ(k \leq l)$ for a successor y and of rule 6
After rule 3 we have either $(x, y) : s$ or $(x, y) : s^{\neg}$ for all $|s|$ in k. Whether it is $(x, y) : s$ or $(x, y) : s^{\neg}$ the first two component do not change because we do not add any new assertions regarding x. The third and fifth component also does not change because we do not add any new successors for x. The fourth component might decreases but never increases: By adding assertions we can only increase the number $n_{sc}(x, y)$ which means that $n_{sc}(x) - n_{sc}(x, y)$ decreases. The sixth component of $\psi_S(x)$ decreases because y does not hold the condition of the fifth component any more. Hence $\psi_{S'} \prec \psi_S(x)$.
For any variable z such that $z \neq y$. The tuple $\psi_S(z)$ is unaffected. It can only be affected by the rules if z is a predecessor of y. But by Lemma 1.2 that would mean that $z = x$.
Because y is a successor of x we know by Lemma 1.3 that the first component of $\psi_{S'}(y)$ is smaller than the first component of $\psi_{S'}(x)$ and therefore $\psi_{S'}(y) \prec \psi_{S'}(x)$. Since the first component of $\psi_{S'}(x)$ does not change we also have $\psi_{S'}(y) \prec \psi_S(x)$. We can obtained $\Psi(S')$ from $\Psi(S)$ by deleting $\psi_S(y)$ and replacing $\psi_S(x)$ by the two smaller septuples $\psi_{S'}(x)$ and $\psi_{S'}(y)$.

4. S' is obtained by the application of rule 4 on $(x, y) : s$:
The first and second component remains unchanged. Also the third and fifth component because we do not add new successors. The fourth component can decrease but never increase: By adding an assertion $(x, y) : r, r \in \mathbf{R}$ we can only increase $n_{sc}(x, y)$ and therefore can only decrease the multiset. With a similar reasoning the sixth component can decrease but never increase. The seventh component always decreases because for a successor y there was no positive role name r such that $(x, y) : r$ but afte rthe rule application there is such an assertion. therefore the fifference becomes smaller.
Let z be a variable such that $z \neq y$. The element $\psi_S(z)$ can only change if z is a predecessor of y. But by Lemma 1.2 that means that $z = x$.
We can obtained $\Psi(S')$ from $\Psi(S)$ by replacing $\psi_S(x)$ with the smaller $\psi_{S'}(x)$.

5. S' is obtained by the application of rule 5a on $x : succ(k < l)$, $x : succ(k \leq l)$ or $x : succ(n\, dvd\, l)$ and rule 7:
For a set term s which occurs as $|s|$ in l we introduce a new variable y and add $(x, y) : s$. The first two component of $\psi_S(x)$ remains unchanged. Because we can apply this rule we have $n_{\mathcal{I}(S)}(x, k, S) > n_{\mathcal{I}(S)}(x, l, S)$ and we have no set term s which occurs in k and in l with the same sign. That means that by adding a new successor to l it can never be a successor to k, too. Therefore only $n_{\mathcal{I}(S)}(x, l, S)$ increases which means $n_{\mathcal{I}(S)}(x, k, S) \trianglelefteq n_{\mathcal{I}(S)}(x, l, S)$ decreases and hence also the third component.
In S' exists now a new tuple $\psi_{S'}(y)$. But since it was introduced by the assertion $x : succ(c)$, $c \in \{k < l, k \leq l, n\, dvd\, l\}$, the first component of it is always smaller then the first component of $\psi_S(x)$.

For any variable z such that $z \neq y$. The tuple $\psi_S(z)$ is unaffected. It can only be affected by the rules if z is a predecessor of y. But by Lemma 1.2 that would mean that $z = x$.
Altogether $\Psi(S')$ can be obtained from $\Psi(S)$ by replacing $\psi_S(x)$ with the two smaller tuples $\psi_{S'}(x)$ and $\psi_{S'}(y)$.

6. S' is obtained by the application of rule 5b on $x : succ(k < l)$ or $x : succ(k \leq l)$:
The first and second component of $\psi_S(x)$ remain unchanged. The third component also remains unchanged: Because we can apply rule 5b we have $l \in \mathbb{N}$ and therefore $n_{\mathcal{I}(S)}(x, k, S) > l$ which means the integer in this multiset is 0. By merging two successor we have $n_{\mathcal{I}(S)}(x, k, S) \geq n_{\mathcal{I}(S)}(x, l, S)$ which means the asymmetrical difference $n_{\mathcal{I}(S)}(x, k, S) \trianglelefteq n_{\mathcal{I}(S)}(x, l, S)$ is still 0. The fourth component can decreases: By merging two successor y_1, y_2 the two corresponding entries in the multiset are removed and a new one is added. The new variable y has all assertions of the two successors which means that for some assertions $x : succ(s_1 \subseteq s_2)$, such that $(x, y_1) : s_1 \in S$ and $(x, y_2) : s_2 \in S$ but $(x, y_1) : s_1 \notin S$ and $(x, y_2) : s_2 \notin S$, we have after the rule application $(x, y) : s_1 \in S$ and $(x, y) : s_2 \in S$ which means that $n_{sc}(x, y) > n_{sc}(x, y_1)$ and $n_{sc}(x, y) > n_{sc}(x, y_2)$. Therefore $n_{sc}(x) - n_{sc}(x, y)$ is smaller than $n_{sc}(x) - n_{sc}(x, y_1)$ or $n_{sc}(x) - n_{sc}(x, y_2)$. The fourth component can not increases because that would mean that by merging two successors we had lost assertions regarding y_1 and y_2. The fifth component decreases because we have one successor less and therefore $\psi_{S'}(x)$ is smaller than $\psi_S(x)$. The new tuple $\psi_{S'}(y)$ is also smaller than $\psi_S(x)$ because y has the same assertions of the two merged successors whose first component are always smaller than the first component of $\psi_S(x)$ because of Lemma 1.3.
No other tuples $\psi_S(z)$ are affected because otherwise z must be a predecessor of y and by Lemma 1.2 $z = x$.
Therefore $\Psi(S')$ can be obtained from $\Psi(S)$ by deleting the tuples of the two merged successors and by replacing $\psi(x)$ with the smaller tuples $\psi_{S'}(x)$ and $\psi_{S'}(y)$.

7. S' is obtained by the application of rule 6 on $x : succ(s_1 \subseteq s_2)$ and rule 7:
After rule 6 S contains $(x, y) : s_2$. Then rule 7 is applied until inapplicable. After rule 7 we can have multiple $(x, y) : r$, $r \in \mathbf{R}$, and/or $y : C$. The first and second component do not change. The third component also does not change because we do not add more successors. The fourth component always decreases because the number $n_{sc}(x, y)$ increases. For any $y : C$ $\psi_S(x)$ remains unchanged but we know that the first component of $\psi'_S(y)$ is smaller then the first component of $\psi_S(x)$ by Lemma 1.3.
For any variable z such that $z \neq y$ the tuple $\psi_S(z)$ is unaffected. It can only be affected by the rules if z is a predecessor of y. But by Lemma 1.2 that would mean that $z = x$.
Therefore $\Psi(S')$ can be obtained from $\Psi(S)$ by deleting $\psi_S(y)$ and by replacing $\psi_S(x)$ with the two smaller septuples $\psi_{S'}(x)$ and $\psi_{S'}(y)$.

□

Lemma 3. If the Tableau-algorithm terminates without a clash then S is satisfiable

Proof.
Again the proof is sectioned by the obtained derived sets.
Let $\mathcal{I}(S')$ be the induced interpretation of the constraint set S' created by the Tableau-algorithm from S. We show that $\pi_{\mathcal{I}(S')}$ satisfies S'.
We start with the simple assertions $x : C$ and $(x, y) : r$ for $C \in \mathbf{C}$ and $r \in \mathbf{R}$ (induction base): By the definition of induced interpretation we assign $\pi_{\mathcal{I}(S')}(x) := x^{\mathcal{I}(S')} \in C^{\mathcal{I}(S')}$.
Also by the definition of induced interpretation for every $(x, y) : r \in S'$ we have $(\pi_{\mathcal{I}(S')}(x), \pi_{\mathcal{I}(S')}(y)) := (x^{\mathcal{I}(S')}, y^{\mathcal{I}(S')}) \in r^{\mathcal{I}(S')}$.
Let S be a constraint set and $\pi_{\mathcal{I}(S)}$ be an assignment which satisfies S (induction hypothesis).

1. If we can apply rule 1 and obtain S' then there must be an assignment $x : C_1 \sqcap C_2 \in S$. By the definition of induced interpretation and by the hypothesis we already have $\pi_{\mathcal{I}(S')}(x) \in C_1^{\mathcal{I}(S')}$ and $\pi_{\mathcal{I}(S')}(x) \in C_2^{\mathcal{I}(S')}$. By adding $x : C_1$ and $x : C_2$ we do not change $\mathcal{I}(S)$. Hence $\mathcal{I}(S') := \mathcal{I}(S)$ and $\pi_{\mathcal{I}(S')} := \pi_{\mathcal{I}(S)}$ satisfies S'.

2. If we can apply rule 2 and obtain S' then there must be an assignment $x : C_1 \sqcup C_2 \in S$. Like above by the definition of the induced interpretation we have $\pi_{\mathcal{I}(S')}(x) := x^{\mathcal{I}(S')}$. We also know that $x^{\mathcal{I}(S')}$ is in $C_1^{\mathcal{I}(S')} \cup C_2^{\mathcal{I}(S')}$. By adding either $x : C_1$ or $x : C_2$ we do not change $\mathcal{I}(S)$. Hence $\mathcal{I}(S') := \mathcal{I}(S)$ and $\pi_{\mathcal{I}(S')} := \pi_{\mathcal{I}(S)}$ satisfies S'.

3. If we can apply rule 3 and obtain S' then we have a constraint $x : \mathit{succ}(k \leq l)$ and a successor y such that $(x, y) : k' \in S$, k' occurs in k. We then choose between $(x, y) : s$ and $(x, y) : s^{\neg}$ for all $|s|$ in k then apply rule 7 until this rule is inapplicable. That means at the end we might add several assertions of the form $x : C$ and $(x, y) : r$. In case we add $x : C$ we also add $x^{\mathcal{I}(S')}$ to $C^{\mathcal{I}(S')}$. Therefore in this case $\pi_{\mathcal{I}(S')}$ satisfies S'. In case we add $(x, y) : r$ we also add $(x^{\mathcal{I}(S')}, y^{\mathcal{I}(S')})$ to $r^{\mathcal{I}(S')}$. Hence $\pi_{\mathcal{I}(S')}$ satisfies S'.

4. If we can apply rule 4 and obtain S' then we have a constraint $(x, y) : k$ but for every role name r we do not have $(x, y) : r \in S$, where r has a positive sign in this assertion. After adding $(x, y) : r$, $r \in \mathbf{R}$, to S' the element $(x^{\mathcal{I}(S)}, y^{\mathcal{I}(S)})$ is also added to $r^{\mathcal{I}(S')}$. Hence $\pi_{\mathcal{I}(S')}$ satisfies S'.

5. If we can apply rule 5 and obtain S' then we have a constraint $x : \mathit{succ}(c)$ such that it is violated regarding x. We either add a new successor y or merge two successor y_1, y_2 to one successor y.
   In the first case we introduce y and add $(x, y) : l$ to S and then apply rule 7 until this rule is inapplicable. When we introduce y we also add a new element $y^{\mathcal{I}(S')}$ to $\mathcal{I}(S')$. For each $y : C$ we add $y^{\mathcal{I}(S')}$ to $C^{\mathcal{I}(S')}$ and for each $(x, y) : r$, $r \in \mathbf{R}$, we add $(x^{\mathcal{I}(S')}, y^{\mathcal{I}(S')})$ to $r^{\mathcal{I}(S')}$. Therefore let $\pi_{\mathcal{I}(S')} := \pi_{\mathcal{I}(S)} \cup \{y \mapsto y^{\mathcal{I}(S')}\}$.
   In the second case there are two successors y_1 and y_2 for which $(x, y) : s$ and

$(x, y) : s$ are in S. If we merge both together to y we also have to merge $y_1^{\mathcal{I}(S)}$ and $y_2^{\mathcal{I}(S)}$ to one element $y^{\mathcal{I}(S')}$. For each $y_i : C \in S$, $i \in \{1, 2\}$ we have $y_i^{\mathcal{I}(S)} \in C^{\mathcal{I}(S)}$ and for each $(x, y_i) : r$, $r \in \mathbf{R}$ we have $(x^{\mathcal{I}_S}, y_i^{\mathcal{I}(S)}) \in r^{\mathcal{I}(S)}$ due to the hypothesis. That means that by merging both elements the element $y^{\mathcal{I}(S')}$ must be in $C^{\mathcal{I}(S')}$ for every $y_i^{\mathcal{I}(S)} \in C^{\mathcal{I}(S)}$ and the element $(x^{\mathcal{I}(S')}, y^{\mathcal{I}(S')})$ must be in $r^{\mathcal{I}(S')}$ for every $(x^{\mathcal{I}(S)}, y_i^{\mathcal{I}(S)}) \in r^{\mathcal{I}(S)}$. Therefore let $\pi_{\mathcal{I}(S')} := \pi_{\mathcal{I}(S)} \backslash \{y_1 \mapsto y_1^{\mathcal{I}(S)}, y_2 \mapsto y_2^{\mathcal{I}(S)}\} \cup \{y \mapsto y^{\mathcal{I}(S)}\}$ which satisfies S'.

6. If we can apply rule 6 and obtain S' then we have a constraint $x : succ(c_1 \subseteq c_2)$ and a successor y such that $(x, y) : c_1 \in S$ but $(x, y) : c_2 \notin S$. By adding $(x, y) : c_2$ to S we have also to add $y : C$ for every concept C in c_2 and $(x, y) : r$ for every role name r in c_2. That means that $x^{\mathcal{I}(S)}$ is added to every $C^{\mathcal{I}(S)}$ and that $(x^{\mathcal{I}(S)}, x^{\mathcal{I}(S)})$ is added to every $r^{\mathcal{I}(S)}$. Therefore the assignment $\pi_{\mathcal{I}(S')} := \pi_{\mathcal{I}(S)}$ satisfies S'.

□

Lemma 4. If S is satisfiable then the Tableau-algorithm terminates without a clash.

Proof. □

References

[1] B. Hollunder and F. Baader. Qualifying Number Restrictions Concept Languages. Technical report, Research Report RR-91-03, 1991.

[2] S. Tobies. *Complexity Results and Practical Algorithms for Logics in Knowledge Representation.* PhD thesis, Rheinisch-Westfälischen Technischen Hochschule Aachen, 2001.