

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 $k > l$
 - $n \text{ dvd } k$ and $n \neg \text{dvd } k$

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 (*ALCSCC*). Concepts are:

- all concept names
- $\text{succ}(c)$ if c is a set or cardinality constraint over *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 an arbitrary concept. Its *NNF* is obtained by applying the following rules

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

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 an assertion in *NNF* the negation sign can only occur 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 $\mathcal{ALCS\mathcal{CC}}$ 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 $\perp^{\mathcal{I}} = \emptyset^{\mathcal{I}}$
- $(C \sqcap D)^{\mathcal{I}} := C^{\mathcal{I}} \cap D^{\mathcal{I}}, (C \sqcup D)^{\mathcal{I}} := C^{\mathcal{I}} \cup D^{\mathcal{I}}$
- $(\neg C)^{\mathcal{I}} := \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
- $(s \cap t)^{\mathcal{I}} := s^{\mathcal{I}} \cap t^{\mathcal{I}}, (s \cup t)^{\mathcal{I}} := s^{\mathcal{I}} \cup t^{\mathcal{I}}$
- $(s^{\neg})^{\mathcal{I}} := \mathcal{U}^{\mathcal{I}} \setminus s^{\mathcal{I}}$
- $|s|^{\mathcal{I}} := |s^{\mathcal{I}}|$
- $(k + l)^{\mathcal{I}} := (k^{\mathcal{I}} + l^{\mathcal{I}}), (n \cdot k)^{\mathcal{I}} := n \cdot k^{\mathcal{I}}$
- $\text{succ}(c)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \text{the mapping } \cdot^{\mathcal{I}_x} \text{ satisfies } c\}$

The mapping $\cdot^{\mathcal{I}_x}$ maps \emptyset to $\emptyset^{\mathcal{I}_x}$, \mathcal{U} to $\mathcal{U}^{\mathcal{I}_x} := \{\bigcup_{r \in \mathbf{R}} r^{\mathcal{I}_x}(x)\}$, 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 $s^{\mathcal{I}} \subseteq t^{\mathcal{I}}$
- $k \leq l$ iff $k^{\mathcal{I}} \leq l^{\mathcal{I}}$
- $n \text{ dvd } l$ iff $\exists m \in \mathbb{N} : n \cdot m = l^{\mathcal{I}}$

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

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

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

2 Tableau

Similar in [1] and [2], where a variable can be replaced by another variable, we will have to 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:

$$S = \{x : succ(|r \cap A| = 1) \sqcap succ(|r \cap B| = 1) \sqcap succ(|r| > 1), \\ y_1 : A, y_2 : B, x.r.y_1, x.r.y_2\} \quad (1)$$

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 :

$$S = \{x : succ(|r| \leq 1), x.r.y_1, x.r.y_2, \\ y_1 : succ(|s| \leq 1), y_2 : succ(|s| \leq 1), y_1.s.z_1, y_2.s.z_2\} \quad (2)$$

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.

For the Tableau-algorithm we define the 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 $s \supseteq t$

2.1 Restrictions

Dealing with \mathcal{ALCSCC} concept is challenging, especially cardinality constraints. Therefore we restrict cardinality constraint to a more simple form.

Definition 9 (*CNNF*). A constraint is in conjunctive negated normal form (*CNNF*) if no disjunction appears in a cardinality term.

Let k be a cardinality term. Each $|s_1 \cup s_2 \cup \dots \cup s_n|$ in k is split into

$$\begin{aligned} & |s_1 \cap s_2 \cap \dots \cap s_n| + \\ & |s_1^- \cap s_2 \cap \dots \cap s_n| + |s_1 \cap s_2^- \cap \dots \cap s_n| + \dots + |s_1 \cap s_2 \cap \dots \cap s_n^-| \\ & |s_1^- \cap s_2^- \cap \dots \cap s_n| + |s_1^- \cap s_2 \cap s_3^- \dots \cap s_n| + \dots + |s_1 \cap s_2 \cap \dots s_{n-1}^- \cap s_n| \\ & \dots \\ & |s_1 \cap s_2^- \cap \dots \cap s_n^-| + |s_1^- \cap s_2 \cap \dots \cap s_n^-| + \dots + |s_1^- \cap s_2^- \cap \dots \cap s_n| \end{aligned}$$

We want to split $|s_1 \cup s_2 \cup \dots \cup s_n|$ into set terms such that they are disjoint to each other. This helps to determinate all properties of successors in k .

2.2 Algorithm

To maintain readability we write $k \leq l$ instead of $l \geq k$ and $k < l$ instead of $l > k$. Therefore $k \lesseqgtr l$ can only represent $k \leq l$, $k = l$ or $k < l$ from now on.

Like in [1] and [2] we have to be *safe* when introducing new variables otherwise we may end in a endless loop or with a false output.

Let $x : succ(k \leq l)$ or $x : succ(k < l)$, where l has at least one cardinality term $|s|$, be violated regarding x in an interpretation \mathcal{I} . It is *safe* to introduce a new variable y and add $(x, y) : l$ to $S' := S \cup \{(x, y) : l\}$ if

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

Same goes for $(x, y) : k$. This says that it is only safe to add a variable if the number of successor in l increases faster then the number of successors in k .

Definition 10 (Tableau). Let S be a set of assertions in *CNNF*.

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 $x : succ(k < l)$ or $x : succ(k \leq l)$ and x has a successor y such that for some for some concept C in k we have $(x, y) : C \in S$ but $(x, y) : k \notin S$
 \rightarrow either $S := S \cup \{(x, y) : k\}$ or $S := S \cup \{(x, y) : k^-\}$
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*₁-rule: S contains $x : succ(c)$, with $c \in \{k \leq l, k < l\}$, such that c is violated in $\mathcal{I}(S)$ regarding x . Choose either $i := k$ or $i := l$, such that i contains at least one cardinality term $|s|$. If it is safe to add a variable y in i then
 \rightarrow choose a set term s , such that $|s|$ is in i , and $S := S \cup \{(x, y) : s\}$, then jump to rule 8
6. *cardinality*₂-rule: S contains $x : succ(n \text{ d}vd l)$, which is violated in $\mathcal{I}(S)$ regarding x . If 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 by merging them no satisfied constraint regarding x becomes violated, then:
 \rightarrow merge y_1 and y_2
7. *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 8
8. *set.term*-rule (Repeat until inapplicable): In S is $(x, y) : s$ and
 - a) $s = s_1 \cap s_2$ but $\{(x, y) : s_1, (x, y) : s_2\} \not\subseteq S$
 $\rightarrow S := S \cup \{(x, y) : s_1, (x, y) : s_2\}$
 - b) $s = s_1 \cup s_2$ and neither $\{(x, y) : s_1\} \subseteq S$ nor $S \setminus \{(x, y) : s_2\} \subseteq S$
 \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 $\mathcal{ALCCSCC}$ concepts
 $\rightarrow S := S \cup \{y : C\}$

Note that:

- s in 5 can also be of the form t^\neg .
- if $n_1 \text{ d}vd n_2 \cdot l$ and $\text{mod}(n_2, n_1) \neq 0$ then $n_1 \text{ d}vd l$ eventually

Definition 11 (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 12 (Derived Set). A *derived set* is a constraint set S' where rule 8 is not applicable.

The first rule decompose the conjunction and the second rule adds non-deterministically the right assertion. The *choose*-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 :

Example 1.

$$S = \{x : succ(|r \cup s| > 1), (x, y) : r\}$$

However there might be an interpretation \mathcal{I}' where y is also a s -successor of x 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 2.

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

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. First we apply rule 5 to actually add a successor. Therefore y is introduced (it is safe because $n_{\mathcal{I}(S)}(x, r^\neg, S)$ increases) and $(x, y) : r^\neg$ is added to S . Now no more rules are applicable except for rule 4. With that rule we can pick either r or s . We can not pick r because r^\neg occurs in the assertion. Therefore we have to pick s .

In rule 5 we first make a choice whether we take k or l from a cardinality constraints $k \lesseqgtr l$, where the chosen cardinality term has at least a term of the form $|s|$ with s being a set term. Intuitively it would be better to only want to increase l but there are cases, where we have to add a successor in k instead.

Example 3.

$$\begin{aligned} \mathbf{R} &= \{\mathbf{r}, \mathbf{s}\} \\ S &= \{x : succ(|s \cap r| + |s \cap r^\neg| + |s^\neg \cap r| < |s| + |r|)\} \end{aligned}$$

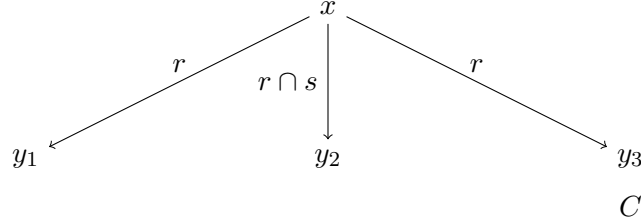
Note that $|s \cap r| + |s \cap r^\neg| + |s^\neg \cap r| = |s \cup r|$. No matter how often we add a successor in $|r|$ or $|s|$, the left hand side always increases and therefore this assertion stays violated. However if we add a successor in $|s \cap r|$ the assertion becomes satisfied. Hence for this example we can choose rule 5 and then choose $i = |s \cap r| + |s \cap r^\neg| + |s^\neg \cap r|$. We then introduce y and add $(x, y) : s \cap r$ to S . Also note that we can not choose $i = |s| + |r|$ because it is not safe: By adding $(x, y) : s$ or $(x, y) : r$ to S the number on the right hand side of the cardinality term does not increases faster then the left hand side.

We consider now two examples to explain rule 6

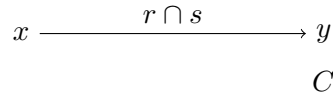
Example 4. Consider the following example

$$\begin{aligned} S &= \{x : succ(|r| = 1) \sqcap succ(|r \cap s| = 1) \sqcap succ(|r \cap C| = 1)\} \\ &\quad x : succ(|r| = 1), x : succ(|r \cap s| = 1), x : succ(|r \cap C| = 1)\} \end{aligned}$$

By applying rule 5 and add a new variable y we satisfies at least one assertion from the second line but still at least one assertion remains violated. If we apply it a second time to satisfy the remaining assertion(s) then $x : succ(|r| = 1)$, which was satisfied before, becomes violated. In case we decided to apply 5 on $x : succ(|r| = 1)$ and $x : succ(|r \cap s| = 1)$ we have the option of either applying rule 5 on $x : succ(|r \cap C| = 1)$ or applying rule 6 on the two new variables. We decide for the first choice:



In this case we have to merge three variables: y_1 , y_2 and y_3 . We can apply 6 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 .



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

Example 5.

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

First we apply the rules 5 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 6 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 5 again leading to $x : succ(|r| < 2)$ being violated and so on. By the condition in 6 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 8 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 \perp
- 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 13. 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 $n_{\mathcal{I}(S)}(x, k, S) \leq n_{\mathcal{I}(S)}(x, l, S)$
- 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 6 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 5 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 $size(succ(k \leq l)) > size(D)$. 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 than $\max\{size(C) \mid x : C \in S\}$.

□

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 replaced 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 increase and hence the component also might decrease. 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 7

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 decrease 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 obtain $\Psi(S')$ 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 after the rule application there is such an assertion. therefore the difference 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 obtain $\Psi(S')$ from $\Psi(S)$ by replacing $\psi_S(x)$ with the smaller $\psi_{S'}(x)$.

5. S' is obtained by the application of rule 5 on $x : succ(k < l)$, $x : succ(k \leq l)$ or $x : succ(n \text{ d}vd l)$ and rule 8:

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 components of $\psi_S(x)$ remain 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 $n_{\mathcal{I}(S)}(x, k, S) \leq n_{\mathcal{I}(S)}(x, l, S)$ 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 $x : succ(c)$, $c \in \{k < l, k \leq l, n \text{ d}vd l\}$, the first component of it is always smaller than 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 6 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 6 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 decrease: 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 increase 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 7 on $x : succ(s_1 \subseteq s_2)$ and rule 8:
 After rule 7 S contains $(x, y) : s_2$. Then rule 8 is applied until inapplicable. After rule 8 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 than 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 $x : \text{succ}(k \leq l)$ 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 8 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 ?? and obtain S' then we have a constraint $x : \text{succ}(c)$ 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 8 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 $\pi_{\mathcal{I}(S')} := \pi_{\mathcal{I}(S)} \setminus \{y_1 \mapsto y_1^{\mathcal{I}(S)}, y_2 \mapsto y_2^{\mathcal{I}(S)}\} \cup \{y \mapsto y^{\mathcal{I}(S)}\}$ which satisfies S' .

6. If we can apply rule 7 and obtain S' then we have a constraint $x : \text{succ}(c_1 \subseteq c_2)$ 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.