1 Preliminaries

Let **C** be a set of concept names and **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 $k \not< l$
  - $k \leq l$ and $k \not\leq l$
  - $n\ dvd\ k$ and $n\ \neg dvd\ k$

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

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

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

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. Let C be a arbitrary concept. Its NNF is obtained by applying the following rules

- $\neg\top \rightarrow \bot$
- $\neg\bot \rightarrow \top$
- $\neg\neg C \rightarrow C$
- $\neg(C \sqcap D) \rightarrow \neg C \sqcup \neg D$
- $\neg(C \sqcup D) \rightarrow \neg C \sqcap \neg D$
- $\neg succ(c) \rightarrow succ(\neg c)$
- $\neg(k < l) \rightarrow k \geq l$
- $\neg(k \not< l) \rightarrow k < l$
- $\neg(n\ dvd\ k) \rightarrow n\ \neg dvd\ k$
- $\neg(n\ \neg dvd\ k) \rightarrow n\ dvd\ k$

With $NNF(C)$ we denote the concept which is obtained by applying the rules above on C until none is applicable any more.
The set S is a set of constraints 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 a set of variables occurring in S.

Definition 4 (Interpretation)**.** An *interpretation* $\mathcal{I} = (\cdot^{\mathcal{I}}, \cdot^{\mathcal{I}})$ over $\mathcal{ALCSCC}$ consists of a non-empty set $\Delta^{\mathcal{I}}$ and a mapping $\cdot^{\mathcal{I}}$ which maps:

- every concept names $A \in \mathbf{C}$ to $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
- every role name $r \in \mathbf{R}$ to $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$

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

- $x : C$ iff $\pi_{\mathcal{I}}(x) \in C^{\mathcal{I}}$
- $(x, y) : s$ iff for every role name r in s $(\pi_{\mathcal{I}}(x), \pi_{\mathcal{I}}(y)) \in r^{\mathcal{I}}$ and for every concept C in s $\pi_{\mathcal{I}}(y) \in C^{\mathcal{I}}$

A constraint c is satisfiable if there exists an interpretation $\mathcal{I}$ and an assignment $\pi_{\mathcal{I}}$ such that $\pi_{\mathcal{I}}$ satisfies c. $\pi_{\mathcal{I}}$ satisfies a constraint set S if $\pi_{\mathcal{I}}$ satisfies every constraint in S. S is satisfiable if there exists an interpretation $\mathcal{I}$ and an assignment $\pi_{\mathcal{I}}$ such that $\pi_{\mathcal{I}}$ satisfies S.

For the sake of readability we say "x satisfies C" instead of "*there exists an interpretation $\mathcal{I}$ and an assignment $\pi_{\mathcal{I}}$ such that $\pi_{\mathcal{I}}$ satisfies $x : C$*". Analogue with "(x, y) satisfies s".

Definition 5 (Number of successors)**.** Let S be a set of constraints, x be a variable and k be a cardinality term. The number of successors counting in k is counted recursively:

$$
\begin{aligned}
n_k(x_S) = &\\
&\text{if } k = n && n\\
&\text{if } k = |s| && |\{y | (x, y) : s \in S\}|\\
&\text{if } k = i + j && n_i(x_S) + n_j(x_S)\\
&\text{if } k = n \cdot l && n \cdot n_l(x_S)
\end{aligned}
$$

where n is an integer, s a set term and i, j, l cardinality terms.
A constraint regarding a variable x is *violated* if

- $x : succ(k \leq n)$ and $n < n_k(x_S)$
- $x : succ(k \geq n)$ and $n > n_k(x_S)$
- $x : succ(k \leq l)$ and $n_l(x_S) < n_k(x_S)$
- $x : succ(k \geq l)$ and $n_l(x_S) > n_k(x_S)$
- $x : succ(n\, dvd\, k)$ and $mod(n_k(x_S), n) \neq 0$

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

2 Tableau

For the algorithm we assume that the constraint in the constraint set are in NNF.

Definition 6 (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 constraints 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 constraint $x : succ(|r| > 1)$ which was satisfied becomes violated.
But not only constraints 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 constraint is violated and therefore merging y_1 and y_2 to y would solve the problem but on the other hand the constraints 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 7 (Clash). A constraint set S contains a *clash* if

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

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

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

Definition 8 (Tableau)**.** Let S be a set of constraints.

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 every $|s|$ occurring in k
   $\rightarrow$ for all $|s|$, in which r occurs, either $S := S \cup \{(x, y) : s\}$ or $S := S \cup \{(x, y) : s^{\neg}\}$ and then jump to rule 4a
4. *cardinality*-rule: S contains $x : succ(c)$, with $c \in \{k \leq l,\ k < l,\ n\, dvd\, l\}$, which is violated regarding x
   a) if there is a positive set term s in l
      $\rightarrow$ introduce new variable y and $S := S \cup \{(x, y) : s\}$, then jump to rule 4a
   b) if $l \in \mathbb{N}$ does not contain a set term then merge 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 $(x, y_2) : s \in S$ if no other constraints regarding x become violated
5. *set*-rule: S contains $x : succ(c_1 \subseteq c_2)$ and $(x, y) : c_1$ but not $(x, y) : c_2$
   $\rightarrow S := S \cup \{(x, y) : c_2\}$ and then jump to rule 4a
6. *set.term*-rule (Repeat until inapplicable): In S is $(x, y) : s$ and
   a) $s = s_1 \cap s_2$ but $\{(x, y) : s_1,\ (x, y) : s_2\} \nsubseteq S$, then
      $\rightarrow S := S \cup \{(x, y) : s_1,\ (x, y) : s_2\}$
   b) $s = s_1 \cup s_2$ and neither $\{(x, y) : s_1\} \subseteq S$ nor $S\{(x, y) : s_2\} \subset S$, then
      $\rightarrow$ 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, then
      $\rightarrow S := S \cup \{y : C\}$

Note that:

- $4b$ is never applicable for $n\, dvd\, l$
- $n_1\, dvd\, n_2 \cdot l$ and $mod(n_2, n_1) \neq 0$ then $n_1\, dvd\, l$ eventually

Definition 9 (Derived Set)**.** A *derived set* is a constraint set S' if it was obtained by applying either

- rule 1, 2 or 4b
- rule 3 and 6 until inapplicable
- rule 4a and 6 until inapplicable
- rule 5 and 6 until inapplicable

on a derived set S. Any untouched constraint set is a derived set, too.

The first rule decompose the conjunction and the second rule adds non-deterministically the right constraint. 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 $n_k(x_S)$ to count the successors of x in k which is important for detecting and avoiding violations of constraints. Now there might be a successor y which satisfies only some part of k in the given S such that $n_k(x_S)$ does not count y. However there might be an interpretation $\mathcal{I}$ and the assignment $\pi_{\mathcal{I}}$ where (x, y) satisfies a set term s which occurs as $|s|$ in k. This means we have

$$n_k(x_S) < \{y|\pi_{\mathcal{I}} \text{ satisfies } (x,y) : s \text{ where } |s| \text{ occurs in } k\}$$

This means $n_k(x_S)$ does not count correctly in this interpretation. If there is a constraint $x : succ(k < n)$ then $n > n_k(x_S)$ does not mean for $\mathcal{I}$ that we still have less than n successors in k hence this constraint might end up being violated in this interpretation what we want to avoid. Therefore the third rule adds non-deterministically the fact that y is indeed a successor in k or not which are the only two options for y.
We consider now two examples to explain the rule. 4b

Example 1. Consider the following example

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

If we try to satisfy at least two of the new constraints by the Tableau-algorithm above we end up with at least one constraint being violated. Let say we use the rules on the three new constraints sequentially. Then we have

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

After using rule 4b two times we have the variable x and its only $r \cap s$-successor y which is of the concept C. We could use this rule because we do not violate any other constraints.

This condition helps to prevent an infinite chain of rule application:

Example 2.

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

First we apply the rules 4a and 6c two times to add two r-successors for x hence $x : succ(|r| < 2)$ is not satisfied any more. If we ignore the condition in rule 4b 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 apply the rules 4a and 6c again leading to $x : succ(|r| < 2)$ being violated again and so on. By the condition in 4b we can not use the rule which means the algorithm terminates with a clash stating that the constraint set is unsatisfiable.

We also know that the application of rule 6 eventually terminates because **C** and **R** are finite and therefore also our constraints.
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. The elements from Q can be ordered by a strict partial order $\prec$. One property of a strict partial order is that it is well-founded e.g. there is no infinite decreasing chain. We can show that the algorithm terminates if for any derived set S' obtained from a derived set S we have $S' \prec S$.
The elements in Q are finite multisets of quintuples and the elements of the quintuples are either integers or mutlisets of integers. For two quintuples $q = (q_1, \ldots, q_5)$ and $q' = (q'_1, \ldots, q'_5)$ it holds $q \prec q'$ if for the first i, $1 \leq i \leq 5$, 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.
Two more definition are needed: Two non-negative integer n, m have the asymmetrical difference $n - m$ if $n \geq m$ and 0 if $n < m$. It is denoted as $n \dot{-} m$. For a concept C its size $|C|$ is inductively defined as

- 1, if C is a primitive concept of **C**
- $|\neg C| = |C|$
- $|succ(c)| = 1 + \sum_{C \in \mathbf{C} \text{ occurs in c}} |C|$
- $|C \sqcap D| = |C \sqcup D| = |C| + |D|$

The quintuples in Q are defined as follows

Definition 10. Let S be a constraint set. The multiset $\Psi(S)$ consist of quintuples $\psi_S(x)$ for each variable x. The component of the quintuples are structured as follows

- the first component is a non-negative integer $max\{|C| \mid x : C \in S\}$
- 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 $|C \sqcap D|$ (respectively for $C \sqcup D$)
- the third component is a multiset of integers containing for each $x : succ(n \leq l) \in S$ the non-negative integer $n \dot{-} n_l(x_S)$ and for each $x : succ(k \leq l) \in S$ the non-negative integer $n_k(x_S) \dot{-} n_l(x_S)$
- the fourth component denotes the number of all successors of x in S
- the fifth 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 for a role r we have $(x, y) : r$, r occurs in k but for at least one $|s|$ in k we have neither $(x, y) : s \in S$ nor $(x, y) : \neg s \in S$

Lemma 1. The following properties hold

1. For any concept C we have $|C| \geq |NNF(\neg C)|$
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 $r \in \mathbf{R}$ then

$$max\{|C| \mid x : C \in S\} > max\{|D| \mid y : D \in S\}$$

Proof.

1. By induction over the number of applications to compute the negation normal form we have $|C| = |NNF(\neg C)|$. Because $\neg succ(k \geq 0)$ can be replace by $\perp$ which is *smaller* than $\neg succ(k \geq 0)$, we have $|C| \geq |NNF(\neg C)|$. This can be done because $\neg succ(k \leq 0) \rightarrow succ(k < 0)$ which is impossible to satisfy.
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 4b 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 4a then we have $y : C$ for every positive concept C occurring in l. But by the definition of the concept size $max\{|C| \mid x : C \in S\}$ is greater by at least 1. A new constraint $y : D$ can occur either because rule 1 or 2 are applicable on $y : C$, which neither raise $max\{|D| \mid y : D \in S\}$, or because rule 3 is applicable on $x : C$, which means that $C = succ(c)$ and therefore it remains $max\{|C| \mid x : C \in S\} > max\{|D| \mid y : D \in S\}$. 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 $max\{|C| \mid x : C \in S\}$.

□

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

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

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

Proof.

1. S' is obtained by the application of rule 1 on $x : C \sqcap D$:
   The first component remains the same because $|C| < |C \sqcap D|$ and $|D| < |C \sqcap D|$. The second component is reduced 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 or a conjunction then the second components becomes smaller because $|C|$ and $|D|$ are always smaller than $|C \sqcap D|$.
   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. More precisely the third and fifth component of $\psi_S(y)$ may be affected. But by adding the constraints $x : C$ and/or $x : D$ the third and fifth component of $\psi_S(y)$ can only become smaller.
   This means, we can obtain $\Psi(S')$ from $\Psi(S)$ by at least 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 4b on $x : succ(k < l)$ or $x : succ(k \leq l)$:
   The first and second component of $\psi_S(x)$ remain unchanged. Because it holds that $n_k(x_S) > l$ we have $l \dot{-} n_k(x_S) = 0$. By merging two successor of x to derive S' we have $n_k(x_S) \leq n_k(x_{S'})$ and therefore $l \dot{-} n_k(x_{S'}) = 0$ which means the third component remains unchanged. The fourth component however 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 constraints of the two merged successors which means the first component is smaller than the first one if $\psi_S(x)$.
   No other tuples $\psi_S(z)$ are affected because otherwise z must be a predecessor of y and by Lemma 1 $z = x$.

4. S' is obtained by the application of rule 3 on $x : succ(k < l)$ for a successor y and of rule 5
   After rule 3 we have either $(x, y) : s$ or $(x, y) : s^{\neg}$ for all $|s|$ in k. If all added constraints are negated the first four component of $\psi_S(x)$ remains unchanged but the fifth component decreases. We look now at an arbitrary positive constraint $(x, y) : s$: We have to apply the rule 6 until it is inapplicable. We know that at some point no more rules are inapplicable because $\mathbf{C}$ and $\mathbf{R}$ are finite and the constraints are in NNF. After rule 6 we can have multiple $(x, y) : r$, $r \in \mathbf{R}$, and/or $y : C$. For any $(x, y) : r$ the third component of $\psi_S(x)$ might be decreased

but never increased. 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 $\psi_S(x)$. In both cases the fifth component of $\psi_S(x)$ decreases by 1 because for each $|s|$ in k we have either $(x, y) : s$ or $(x, y) : s^{\neg}$. So we can obtained $\Psi(S')$ from $\Psi(S)$ by deleting $\psi_S(y)$ and replacing $\psi_S(x)$ by the two smaller quintuples $\psi_{S'}(x)$ and $\psi_(S')(y)$.

5. S' is obtained by the application of rule 4a on $x : succ(k < l)$, $x : succ(k \leq l)$ or $x : succ(n\,dvd\,l)$ and rule 6:
   For a positive 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. The third component of $\psi_S(x)$ must become smaller. It can not remain unchanged because this would mean that $n_l(x_S) \dot{-} n_l(x_S) = 0$ which denotes that we have already at least as many successor in k as in l and therefore would not have been able to apply this rule.
   In S' exists now a new tuple $\psi_{S'}(y)$. But since it was introduced by the constraint $x : succ(c)$, $c \in \{k < l, k \leq l, n\,dvd\,l\}$, the first component of it is always smaller then the first component of $\psi_S(x)$.
   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 5 on $x : succ(c_1 \subseteq c_2)$ and rule 6:
   After rule 5 S contains $(x, y) : c_2$. maybe add more component?

$\square$