

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

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
- $succ(c)$  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. Let  $C$  be a arbitrary concept. Its *NNF* is obtained by applying the following rules

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

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

The 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  $S$ .

**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:

- $\emptyset$  to  $\emptyset^{\mathcal{I}}$
- $\mathcal{U}$  to  $\mathcal{U}^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
- 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

- $(C \sqcap D)^{\mathcal{I}} := C^{\mathcal{I}} \cap D^{\mathcal{I}}, (C \sqcup D)^{\mathcal{I}} := C^{\mathcal{I}} \cup D^{\mathcal{I}}$
- $(\neg C)^{\mathcal{I}} := \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
- $(s \cap t)^{\mathcal{I}} := s^{\mathcal{I}} \cap t^{\mathcal{I}}, (s \cup t)^{\mathcal{I}} := s^{\mathcal{I}} \cup t^{\mathcal{I}}$
- $(s^\neg)^{\mathcal{I}} := \mathcal{U}^{\mathcal{I}} \setminus s^{\mathcal{I}}$
- $|s|^{\mathcal{I}} := |s^{\mathcal{I}}|$

- $(k + l)^{\mathcal{I}} := (k^{\mathcal{I}} + l^{\mathcal{I}}), (n \cdot k)^{\mathcal{I}} := n \cdot k^{\mathcal{I}}$
- $\text{succ}(c)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \text{the mapping } \cdot^{\mathcal{I}_x} \text{ satisfies } c\}$

The mapping  $\cdot^{\mathcal{I}_x}$  maps  $\emptyset$  to  $\emptyset^{\mathcal{I}}$ ,  $\mathcal{U}$  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  $r^{\mathcal{I}_x} := r^{\mathcal{I}}$ .

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

An *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  $(\pi_{\mathcal{I}}(x), \pi_{\mathcal{I}}(y)) \in s^{\mathcal{I}}$

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  $S$ .

**Definition 5** (Number of successors). Let  $S$  be a set of constraints,  $\mathcal{I}$  be an interpretation,  $\pi_{\mathcal{I}} : \text{Var}(S) \rightarrow \Delta^{\mathcal{I}}$  be an assignment,  $x$  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  $n_{\mathcal{I}}(x_{S_k}) := |k^{\mathcal{I}} \cap r^{\mathcal{I}}(x)|$ .

A constraint regarding a variable  $x$  in an interpretation  $\mathcal{I}$  is *violated* if

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

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

## 2 Tableau

For the algorithm we assume that the constraints 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:

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

If we merge  $y_1$  and  $y_2$  then the constraint  $x : succ(|r| > 1)$  which was satisfied becomes violated.

But not only constraints regarding  $x$  might become violated after merging two successors of  $x$ :

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

We see that the first 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  $z_2$ .

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  $s \supseteq t$

**Definition 7** (Induced Interpretation  $\mathcal{I}_I$ ). An interpretation  $\mathcal{I}_I$  can be induced from a constraint set  $S$  by the following steps:

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

With  $\mathcal{I}_I$  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:

- It is safe to add a new variable if there is no set term  $s$  which occurs in  $k$  and  $l$  with the same sign
- It is safe to merge two variable if by merging them we do not violate any constraint regarding the predecessor in the induced interpretation of the resulting constraint set

**Definition 8** (Tableau). Let  $S$  be a set of constraints 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  $|s|$  occurring 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 6
4. *cardinality*-rule:  $S$  contains  $x : succ(c)$ , with  $c \in \{k \leq l, k < l, n \text{ d}vd l\}$ , such that  $c$  is violated in  $\mathcal{I}_I$  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 6
  - b) if  $l \in \mathbb{N}$  does not contain a set term  
 $\rightarrow$  if it is safe to 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  $(x, y_2) : s \in S$  then merge them
5. *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 6
6. *set.term*-rule (Repeat until inapplicable): In  $S$  is  $(x, y) : s$  and
  - a)  $s = s_1 \cap s_2$  but  $\{(x, y) : s_1, (x, y) : s_2\} \not\subseteq S$ , 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  $S \setminus \{(x, y) : s_2\} \subseteq S$ , 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  $\mathcal{ALCS\mathcal{CC}}$  concepts, then  
 $\rightarrow S := S \cup \{y : C\}$

Note that:

- $s$  in 4a can also be of the form  $t^\neg$ .
- 4b is never applicable for  $n \text{ d}vd l$
- 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 9** (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.

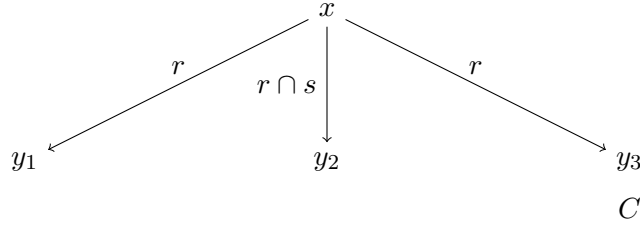
**Definition 10** (Derived Set). A *derived set* is a constraint set  $S'$  without clashes where only rule 6 is not applicable.

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  $n_{\mathcal{I}}(x_{S_k})$  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  $n_{\mathcal{I}}(x_{S_k})$  does not count  $y$ . However there might be an interpretation  $\mathcal{I}'$  and the assignment  $\pi_{\mathcal{I}'}$  such that  $n_{\mathcal{I}'}(x_{S_k})(x, y)$  counts  $y$ . **todo....** We consider now two examples to explain the rule. 4b

**Example 1.** Consider the following example

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

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. We use the rules on the three new constraints sequentially. Then we have



After using rule 4b two times we have the variable  $x$  and its only  $r \sqcap s$ -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 the number of concept names and role names are finite in  $S$  (since  $S$  is finite). 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 sextuples and the elements of the sextuples are either integers or mutlisets of integers. For two sextuples  $q = (q_1, \dots, q_6)$  and  $q' = (q'_1, \dots, q'_6)$  it holds  $q \prec q'$  if for the first  $i$ ,  $1 \leq i \leq 6$ , 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 mutlisets 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 primitive concept of  $\mathbf{C}$
- $size(\neg C) = size(C)$
- $size(succ(c)) = 1 + \sum_{C \in \mathbf{C} \text{ occurs in } c} size(C)$
- $size(C \sqcap D) = size(C \sqcup D) = size(C) + size(D)$

The number  $n_{sc}(x)$  denotes the number of constraints 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 constraints 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 sextuples in  $Q$  are defined as follows

**Definition 11.** Let  $S$  be a constraint set. The multiset  $\Psi(S)$  consist of sextuples  $\psi_S(x)$  for each variable  $x$ . The component of the sextuples 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 of integers in which for each successor  $y$  of  $x$  we have  $n_{sc}(x) - n_{sc}(x, y)$

- the fourth component is a multiset which denotes for every  $x : succ(k \leq l)$  the integer  $n_{\mathcal{I}_l}(x_{S_k}) \leq n_{\mathcal{I}_l}(x_{S_l})$
- 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  $(x, y) : \neg s \in S$

**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 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 applying 4a on a constraint  $x : succ(k \leq l)$  then we have  $y : C$  for every concept  $C$  occurring in  $l$  (for  $\neg C$  we have  $y : \neg C$ ). We know that  $size(succ(k \leq l))$  is greater then  $size(C) =: max\{size(D) \mid y : D \in S\}$  therefore Lemma 1.3 holds. A new constraint  $y : D$  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 constraint  $y : D$  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 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 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 constraint for  $x$  the number  $n_{sc}(y, x)$  might increase and hence the component decreases. The fourth, fifth and sixth component also remain unchanged because the number of  $y$ 's successors does not change. The sixth also does not change because we do not add a constraint 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 4b 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 4b we have  $l \in \mathbb{N}$  and therefore  $n_{\mathcal{I}_I}(x_{S_k}) > l$  which means the integer in this multiset is 0. By merging two successors we have  $n_{\mathcal{I}_I}(x_{S_k}) \geq n_{\mathcal{I}_I}(x_{S_l})$  which means the asymmetrical difference  $n_{\mathcal{I}_I}(x_{S_k}) \triangle n_{\mathcal{I}_I}(x_{S_l})$  is still 0. The fourth 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 constraints 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  $\psi_{S'}(y)$ .

4.  $S'$  is obtained by the application of rule 3 on  $x : succ(k \leq l)$  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  $k$ . 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 constraint regarding  $x$ . The third and fourth component also does not change because we do not add any new successors for  $x$ . The fifth 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 sextuples  $\psi_{S'}(x)$  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  $x : succ(n \text{ d}vd l)$  and rule 6:

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  $n_{\mathcal{I}_I}(x_{S_k}) > n_{\mathcal{I}_I}(x_{S_l})$  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  $n_{\mathcal{I}_I}(x_{S_l})$  increases which means  $n_{\mathcal{I}_I}(x_{S_k}) \trianglelefteq n_{\mathcal{I}_I}(x_{S_l})$  decreases.

In  $S'$  exists now a new tuple  $\psi_{S'}(y)$ . But since it was introduced by the constraint  $x : succ(c)$ ,  $c \in \{k < l, k \leq l, n \text{ d}vd l\}$ , 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 5 on  $x : succ(s_1 \subseteq s_2)$  and rule 6:

After rule 5  $S$  contains  $(x, y) : s_2$ . Then rule 6 is applied until inapplicable. After rule 6 we can have multiple  $(x, y) : r$ ,  $r \in \mathbf{R}$ , and/or  $y : C$ . The first and second component do not change. The fourth component also does not change because we do not add more successors. The fourth component might change but can only decreases: The successor  $y$  might be counted in the fourth component because there exists one  $|s|$  in a cardinality term  $k$  such that  $x : succ(k \leq n) \in S$  and  $(x, y) : s \notin S$  or  $(x, y) : s \in S$ . Also there is already the constraint  $(x, y) : k'$  such that  $k'$  occurs in  $k$ . If  $s = s_2$  then by adding  $(x, y) : s_2$   $y$  does not count any more in the fourth component and hence the fourth component decreases. In any cases the sixth 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 sextuples  $\psi_{S'}(x)$  and  $\psi_{S'}(y)$ .

□