

Contents

1	Introduction	2
2	Preliminaries	3
3	Tableau for \mathcal{ALCSCC}	6
3.1	Transforming an ABox into a formula	8
3.2	Algorithm	8
4	Correctness	9

1 Introduction

Traditional data bases where data are stored solely without any connection towards to themselves like many people would imagine are often not enough any more. The reason is that the data are stored without any semantics. However storing data with semantics can provide additional information. For example we have some data about two objects "Anna" and "Beth". In a traditional data base if not explicitly stated, both data are not related to each other. Nevertheless Anna and Beth can have a relation, which also depends on who or what both are. For example both can be human and Anna is a teacher and Beth is a student. Both are in the same class. By adding solely those information in a traditional data base the information that Anna must teaches Beth is not given. One way to apply semantics to data objects is to use *ontologies*. In biological and (bio)medical researches data bases are often based on ontologies [2]. Ontologies (in the computer science field) can be viewed as formal representation of a certain domain of interest. In data base they are collection of relation between the entities in the data base and are formulated as a fragment of first-order logic (FOL). These fragments of FOL are represented as *Description Logic (DL)*, which is a family of knowledge representation system. DL are mainly built of concepts, which correspond to unary relations in FOL and is often represented by a capital letter, and relation between the concepts, which correspond to binary relations in FOL and is often represented by a lowercase letter. For more complex (compound) concepts operators like \sqcap , \sqcup , \sqsubseteq , \exists and \forall , depending on the DL, are used. For example the statement "All Men and Women are Human" is formalize in FOL as $\forall x. Men(x) \vee Women(x) \rightarrow Human(x)$ and in DL as an *axiom* $Men \sqcup Women \sqsubseteq Human$, where *Men*, *Women* and *Human* are concept names. The statement "All Humans, who have children, are parents" can be formalized in FOL as $\forall x \exists y. Human(x) \wedge hasChildren(x, y) \rightarrow Parent(x)$ and in DL as $Human \sqcap \exists hasChildren. \top \sqsubseteq Parent$, where *Human* and *Parents* are concept names and *hasChildren* is a role name. Restriction with the operators \exists and \forall are called *quantified* restrictions. The second statement can also be formalized with a *qualified* restriction: $Human \sqcap \geq 1 hasChildren. \top \sqsubseteq Parent$. Each quantified restriction can be transformed into a qualified restriction.

One big research field in DL is the determination of satisfiability of an *knowledge base*, which is formulated in DL. A knowledge base normally consists of a *TBox*, which contains the axioms (rules), and of an *ABox* which contains assertions of certain elements (objects). This DL allows conjunctions (\sqcap), disjunctions (\sqcup), negation $\neg C$ and qualifying number restriction ($\leq nr C$ and $\geq nr C$), where n is a number, r a role name, and C a concept name. In [3] a *Tableau*-algorithm is presented for checking satisfiability for an ABox in the DL \mathcal{ALCQ} . A Tableau-algorithm applies *completion rules* to a given *set*(ABox) to decompose complex concepts and try satisfying violated *statements*(assertions). If the set concludes something unsatisfiable (clash) then the whole set is unsatisfiable. If no more rules are applicable and the set is not unsatisfiable, then it is otherwise. The satisfiability (of concepts) is stated in [3] as PSPACE-hard problem (without TBox, with TBox it is EXPTIME-hard [1]. In [5] a optimized Tableau-algorithm is presented which results in a PSPACE-problem. The optimization is that instead of

keeping n successors to satisfy a restriction $\geq nr.C$ like in [3], the algorithm saves the number of existing successors and by comparing the numbers detects possible clashes. This DL is more expressive than \mathcal{ALCQ} because every qualified restriction $\leq nr.C$ and $\geq nr.C$ can be written in \mathcal{ALCSCC} as $\text{succ}(|r.C| \leq 1)$ and $\text{succ}(|r.C| \geq 1)$.

The expressive DL \mathcal{ALCSCC} extends \mathcal{ALCQ} with *set constraint* and *cardinality constraint*, which lays under the logic of $QFBAPA$ (quantifier-free fragment of Boolean Algebra with Presburger Arithmetic). As the name says we do not have quantifier. Instead we use set expression (Boolean Algebra part) and numerical constraint (Presburger Arithmetic) which is combined together with cardinality functions. For example $Human \sqcap \geq 1 \text{ hasChildren} \sqcap \sqsubseteq Parent$ is written in \mathcal{ALCSCC} as $Human \sqcap \text{succ}(|\text{hasChildren}| \geq 1) \sqsubseteq Parent$. In [1] a solution for the satisfiability problem (without TBox) is presented which has the complexity PSpace: For an ABox we guess the value (true or false) of the top-level variables which can already lead to a *false*-result. If not then the constraint is formulated into a $QFBAPA$ formula, for which an algorithm determine by guessing a number N of Venn-region to be non-empty whether the formula is satisfied or not.

In this work we give another solution for the satisfiability problem with a Tableau-algorithm. As in previous work for other DLs we define completion rules which can be applied onto assertions in the ABox to determine whether \perp can be concluded from it which states its unsatisfiability. If we can not apply any rules any more and we can not conclude \perp , then the ABox is satisfiable. The main difficulty is that unlike \mathcal{ALCQ} , where the bond of number of successor is fixed, in \mathcal{ALCSCC} we can compare two cardinalities, which can vary during the algorithm. Hence we need an approach for counting successors and with it calculating the *correct* cardinality, which is necessary to detect satisfied and violated constraint. For this we introduce *induced interpretation* which can determine the cardinalities after each rule application. Further more to deal with the numerical arithmetic of \mathcal{ALCSCC} we use a $QFBAPA$ solver. We transform a subset of the ABox into a $QFBAPA$ formula and then let a solver determine whether the formula is satisfiable or not. If not we end with a clash. If it returns a solution, then we add variables according to it to our ABox.

2 Preliminaries

In this work \mathbf{C} denotes a set of concept names and \mathbf{R} a set of role names, which are disjoint. Before we define the DL \mathcal{ALCSCC} we have to explain first how the language $QFBAPA$ looks like.

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 $n \in \mathbb{N}$

A *QFBABA* formula are disjunction (\vee) and conjunction (\wedge) of (also possible negated) cardinality constraints set variables.

Since $s \subseteq t$ can be expressed as the cardinality constraint $|s \cap t^c| \leq 0$ we will not consider any set constraints further in this work. In case we want to express $x : \text{succ}(s = t)$, with s, t being set terms, we write instead $x : \text{succ}(|s \cap t^c| \leq 0) \sqcap \text{succ}(|s^c \cap t| \leq 0)$. Furthermore instead of $l \geq k$ we write $k \leq l$, instead of $k < l$ we write $k + 1 \leq l$ and instead of $k = l$ we write $k \leq l$ and $l \leq k$. Hence for an assertion $x : \text{succ}(c)$ the cardinality constraint c is either of the form $k \leq l$ or $n \text{ dvd } l$.

The semantic of *QFBAPA* is define as follows:

Definition 2 (Interpretation of *QFBAPA*). Let $\Delta^{\mathcal{I}}$ be a set and σ a mapping which maps

- every symbol a in T to $\sigma(a)$
- \emptyset to $\sigma(\emptyset)$
- \mathcal{U} to $\sigma(\mathcal{U}) \subseteq \Delta^{\mathcal{I}}$
- $\sigma(s \cap t) := \sigma(s) \cap \sigma(t)$, $\sigma(s \cup t) := \sigma(s) \cup \sigma(t)$
- $\sigma(s^c) := \sigma(\mathcal{U}) \setminus \sigma(s)$
- $\sigma(|s|) := |\sigma(s)|$
- $\sigma(k + l) := \sigma(k) + \sigma(l)$, $\sigma(n \cdot k) := n \cdot \sigma(k)$

The mappings satisfies for the cardinality terms k, l

- $k \leq l$ iff $\sigma(k) \leq \sigma(l)$
- $n \text{ dvd } l$ iff $\exists m \in \mathbb{N} : n \cdot m = \sigma(l)$

A solution ϕ for a *QFBAPA* formula is denoted as assignments of elements e to intersections and/or unions of set variables: $\phi(e) = X_1 \cup \dots \cup X_n \cap \dots \cap X_m$. The set $E(\phi)$ denotes the elements, which are assigned from ϕ .

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

Definition 3 (*ALCSCC*). *ALCSCC* concepts are defined inductively:

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

An ABox \mathcal{A} in *ALCSCC* is a finite set of assertions of the form $x : C$ and $(x, y) : r$, where C is a *ALCSCC* concept, $r \in \mathbf{R}$ and x, y are variables. The set $\text{Var}(\mathcal{A})$ is the set of variables occurring in \mathcal{A} .

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

- each variable $x \in \text{Var}(\mathcal{A})$ 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 $\perp^{\mathcal{I}} = \emptyset^{\mathcal{I}}$
- $(C \sqcap D)^{\mathcal{I}} := C^{\mathcal{I}} \cap D^{\mathcal{I}}$, $(C \sqcup D)^{\mathcal{I}} := C^{\mathcal{I}} \cup D^{\mathcal{I}}$
- $(\neg C)^{\mathcal{I}} := \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
- $\text{succ}(c)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \text{the mapping } \cdot^{\mathcal{I}_x} \text{ satisfies } c\}$

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 $r^{\mathcal{I}_x} := r^{\mathcal{I}}(x)$.

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 an ABox \mathcal{A} if $\pi_{\mathcal{I}}$ satisfies every assertion in \mathcal{A} . If $\pi_{\mathcal{I}}$ satisfies \mathcal{A} then \mathcal{I} is a model of \mathcal{A} .

3 Tableau for \mathcal{ALCSCC}

A Tableau-algorithm consist of completion rules to decide satisfiability of a set of assertions. The rules are applied exhaustively on the set until none is applicable any more. One major characteristic of this algorithm is that it does not matter in which order the rules are applied. Another characteristic is that it works non-deterministically: In case we have disjunctions we can choose between the concepts in this disjunctions. If a choice ends in a *clash* then we track back to the point where we had to chose and take the other choice instead. If all choices ends in a clash then the ABox is unsatisfiable, otherwise it is satisfiable.

We want to use the Tableau-algorithm to check whether an assertion $x : C$ is satisfiable or not and if it satisfiable we want to create a satisfied ABox from $x : C$.

To help the algorithm we want to avoid nested negation e.g. $\neg(\neg(\neg(A \cup B)))$. Hence we consider all concepts in *negated normal form (NNF)*.

Definition 5 (Negation Normal Form). A \mathcal{ALCSCC} 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 \mathcal{ALCSCC} concept. With $NNF(C)$ we denote the concept which is obtained by applying the rules below on C until none is applicable any more.

- | | |
|--|--|
| <ul style="list-style-type: none"> • $\neg \top \rightarrow \perp$ • $\neg \perp \rightarrow \top$ • $\neg \neg C \rightarrow C$ • $\neg(C \sqcap D) \rightarrow \neg C \sqcup \neg D$ • $\neg(C \sqcup D) \rightarrow \neg C \sqcap \neg D$ • $C^\neg \rightarrow \neg C$ • $\neg succ(c) \rightarrow succ(\neg c)$ | <ul style="list-style-type: none"> • $\neg(k \leq l) \rightarrow l \leq k$ • $\neg(n \text{ dvd } k) \rightarrow n \neg \text{dvd } k$ • $\neg(n \neg \text{dvd } k) \rightarrow n \text{ dvd } k$ • $(s \cap t)^\neg \rightarrow s^\neg \cup t^\neg$ • $(s \cup t)^\neg \rightarrow s^\neg \cap t^\neg$ • $(s^\neg)^\neg \rightarrow s$ |
|--|--|

The rule $C^\neg \rightarrow \neg C$ is necessary because C^\neg can be a result of s^\neg , where s is a set term. It can be transformed into $\neg C$: For every interpretation \mathcal{I} of S we have $(C^\neg)^{\mathcal{I}} = \mathcal{U} \setminus C^{\mathcal{I}}$

and $(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$. Since $\mathcal{U} \subseteq \Delta$ we can conclude that every element in $(C^{\neg})^{\mathcal{I}}$ is also in $(\neg C)^{\mathcal{I}}$.

The first five rules on the left hand side can be applied in linear time [3],[4]. The first four rules on the right hand side, $C^{\neg} \rightarrow \neg C$ and $\neg succ(c) \rightarrow succ(\neg c)$ can also be applied in linear time since we only shift the negation sign. the rule $(s^{\neg})^{\neg} \rightarrow s$ works similarly to $\neg \neg C \rightarrow C$ and the rules $(s \cap t)^{\neg} \rightarrow s^{\neg} \cup t^{\neg}$ and $(s \cup t)^{\neg} \rightarrow s^{\neg} \cap t^{\neg}$ works the similarly to $\neg(C \sqcap D) \rightarrow \neg C \sqcup \neg D$ and $\neg(C \sqcup D) \rightarrow \neg C \sqcap \neg D$ and can also be applied in linear time. Next we introduce *induced interpretation* with which we can count successors of variables after any rule application.

Definition 6 (Induced Interpretation). An interpretation $\mathcal{I}(\mathcal{A})$ can be induced from an ABox S by the following steps:

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

Since we can now denote the number of successor of a variable x we can determine which assertion of the form $x : succ(c)$ are violated.

Definition 7 (Violated assertion). Let \mathcal{A} be a set of assertion, x be a variable, k be a cardinality term and $n \in \mathbb{N}$. An assertion is *violated* if

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

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

Like already mentioned an ABox is unsatisfiable if all choices ends in a clash.

Definition 8 (Clash). An ABox \mathcal{A} contains a *clash* if

- $\{x : \perp\} \subseteq \mathcal{A}$ or
- $\{x : A, x : \neg A\} \subseteq \mathcal{A}$ or
- $\{x : succ(n \neg \text{dvd } l), x : succ(m \text{dvd } l)\} \subseteq \mathcal{A}$, with $n \text{dvd } m$ or $m \text{dvd } n$, or
- $\{(x, y) : r, (x, y) : r^{\neg}\} \subseteq \mathcal{A}$ or
- $\{x : succ(c)\} \subseteq \mathcal{A}$ violated and no more rules are applicable

3.1 Transforming an ABox into a formula

Dealing with numerical arithmetic is challenging and hence we use the help of a *QFBAPA* s. We want to use it whenever we want to add successors for a variable x . For that we collect all *succ*-assertion regarding x first and then transform them into a *QFBAPA* formula for one *nested level*. As example we look at

Example 1 (Example for transforming ABox into *QFBAPA* formula).

$$\mathcal{A} = \{x : \text{succ}(1 \leq |\text{succ}(|A| \leq |B \cap r|)|), x : \text{succ}(|A| \leq |B|), x : C\}$$

with $\mathbf{C} = \{A, B, C\}$ and $\mathbf{R} = \{r\}$ We first gather all *succ*-assertion regarding x together and transform it into a formula by doing the following steps:

- drop all $x : \text{succ}$
- replace all role names r with X_r
- replace all concepts names C with X_C
- replace all $\text{succ}(c)$ with X_c
- connect all formulas with \wedge
- include the conjunct $\mathcal{U} = X_{r_1} \cup \dots \cup X_{r_n}, r_1, \dots, r_n \in \mathbf{R}$

We replace (possible compound) concepts and role names with set variables, for which a solver can assign elements to them. We only replace concepts on one *nested level*. In the example the first assertion tells that x must have at least one successor y which has more successors in $B \cap r$ than in A . The concept $\text{succ}(|A| \leq |B \cap r|)$ is on a different *nested level* then $\text{succ}(|A| \leq |B|)$. The next step would be to introduce variables and the assignments according to the solution the solver gives us. Hence we also introduced variables for the nested *succ*-assertion. Then we can do the procedure from above again for the new variables.

3.2 Algorithm

Finally we can present the Tableau-algorithm for an ABox in *ALCSCC*. We describe above how we handle the numeric arithmetic of *ALCSCC* and that we want to decompose compound concept first. Hence we divide the algorithm in two parts: a boolean part, where the decomposing of compound concepts takes place, and a numerical part, where a part of the ABox is transformed into a *QFBAPA* formula and a solver returns a possible assignment of elements. The boolean part has a higher priority than the numerical part.

Definition 9 (Tableau). Let \mathcal{A} be a set of assertions in *NNF*.
Boolean part:

- \sqcap -rule: \mathcal{A} contains $x : C_1 \sqcap C_2$ but not both $x : C_1$ and $x : C_2$
 $\rightarrow \mathcal{A} := \mathcal{A} \cup \{x : C_1, x : C_2\}$
- \sqcup -rule: \mathcal{A} contains $x : C_1 \sqcup C_2$ but neither $x : C_1$ nor $x : C_2$
 $\rightarrow \mathcal{A} := \mathcal{A} \cup \{x : C_1\}$ or $\mathcal{A} := \mathcal{A} \cup \{x : C_2\}$

Numerical part:

- *successor*-rule: \mathcal{A} contains for a variable x at least one violated assertion of the form $x : succ(c)$:
 - gather all assertion of the form $x : succ(c)$ into a set \mathcal{S}
 - transform \mathcal{S} into a *QFBAPA* formula
- If a *QFBAPA* solver returns *unsatisfiable* then $\mathcal{A} = \mathcal{A} \cup \{x : \perp\}$
 If a *QFBAPA* solver returns *satisfiable* and a solution ϕ then for each $e \in E(\phi)$ in the solution we introduce a new variable y and
- for each $e \in X_C$ we have $\mathcal{A} = \mathcal{A} \cup \{y : C\}$
 - for each $e \in X_c$, c is a cardinality constraint, we have $\mathcal{A} = \mathcal{A} \cup \{y : succ(c)\}$
 - for each $e \in X_r$, $r \in \mathbf{R}$, we have $\mathcal{A} = \mathcal{A} \cup \{(x, y) : r\}$

We pick up Example 1 again but in a form where it is not decomposed yet:

$$\mathcal{A} = \{x : succ(1 \leq |succ(|A| \leq |B \cap r|)|) \sqcap succ(|A| \leq |B|) \sqcap C, (x, y) : B \cap r\}$$

We are able to apply the boolean rules and hence we apply them first and decompose into Example 1. Then we apply the *successor*-rule: We collect every *succ*-assertion regarding x to a set $\mathcal{S} := \{x : succ(1 \leq |succ(|A| \leq |B \cap r|)|), x : succ(|A| \leq |B|)\}$ and convert \mathcal{A} to the *QFBAPA* formula $1 \leq X_{|A| \leq |B \cap r|} \wedge |X_A| \leq |X_B| \wedge \mathcal{U} = \{X_r\}$. Then we let a solver returns an assignment if possible. In this case we see that the formula is satisfiable with $X_{|A| \leq |B \cap r|} = \{f\}$, $X_A = \{\}$, $X_B = \{e\}$ and $X_r = \{e, f\}$. Since we have $\mathcal{U} = \{X_r\}$ every element must be in X_r . That means that every successor is connected to its predecessor by a role name and in our example every successor is a r -successor. We then introduce for e and f two variables y and z and the assertion $y : X_B, (x, y) : r, z : succ(|A| \leq |B \cap r|)$ and $(x, z) : r$ to \mathcal{S} . Then for z we have to apply the *successor*-rule again.

4 Correctness

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

- If no more rules are applicable on a clash-free ABox \mathcal{A} then \mathcal{A} is satisfiable
- If \mathcal{A} is satisfiable then the Tableau-algorithm terminates without a clash
- For every input the Tableau-algorithm terminates

In all proves we assume that the *QFBAPA* solver is correct. First we prove that the algorithm works correctly e.g. we prove the first two points.

Lemma 1. If the Tableau-algorithm terminates without a clash then \mathcal{A} is satisfiable

Proof. Let $\mathcal{I}(\mathcal{A})$ be the induced interpretation of the clash-free ABox \mathcal{A} after the algorithm terminated. If $\pi_{\mathcal{I}(\mathcal{A})}$ satisfies \mathcal{A} then \mathcal{A} is satisfiable.

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 $\pi_{\mathcal{I}(\mathcal{A})}(x) := x^{\mathcal{I}(\mathcal{A})} \in C^{\mathcal{I}(\mathcal{A})}$. Also by the definition of induced interpretation for every $(x, y) : r \in \mathcal{A}$ we have $(\pi_{\mathcal{I}(\mathcal{A})}(x), \pi_{\mathcal{I}(\mathcal{A})}(y)) := (x^{\mathcal{I}(\mathcal{A})}, y^{\mathcal{I}(\mathcal{A})}) \in r^{\mathcal{I}(\mathcal{A})}$.

Let \mathcal{A}' be an ABox and $\pi_{\mathcal{I}(\mathcal{A}')}$ be an assignment which satisfies \mathcal{A}' (induction hypothesis). Let $\mathcal{I}(\mathcal{A})$ be the induced interpretation of the ABox \mathcal{A} created after one rule of the Tableau-algorithm from \mathcal{A}' (induction step).

1. \sqcap -rule was applied to create \mathcal{A} : Since we could apply this rule, we have $x : C_1 \sqcap C_2 \in \mathcal{A}'$. Since \mathcal{A}' is satisfiable we have $\pi_{\mathcal{I}(\mathcal{A}')}(\pi_{\mathcal{I}(\mathcal{A})}(x)) \in (C_1 \sqcap C_2)^{\mathcal{I}(\mathcal{A}')} = C_1^{\mathcal{I}(\mathcal{A}')} \cap C_2^{\mathcal{I}(\mathcal{A}')}$. By adding $x : C_1$ and $x : C_2$ to \mathcal{A} we do not change $\mathcal{I}(\mathcal{A}')$. Therefore $\pi_{\mathcal{I}(\mathcal{A})} := \pi_{\mathcal{I}(\mathcal{A}')}$ satisfies \mathcal{A} .
2. \sqcup -rule was applied to create \mathcal{A} (Analogue to above): Since we could apply this rule, we have $x : C_1 \sqcup C_2 \in \mathcal{A}'$. Since \mathcal{A}' is satisfiable we have $\pi_{\mathcal{I}(\mathcal{A}')}(\pi_{\mathcal{I}(\mathcal{A})}(x)) \in (C_1 \sqcup C_2)^{\mathcal{I}(\mathcal{A}')} = C_1^{\mathcal{I}(\mathcal{A}')} \cup C_2^{\mathcal{I}(\mathcal{A}')}$. By adding $x : C_1$ or $x : C_2$ to \mathcal{A} we do not change $\mathcal{I}(\mathcal{A}')$. Therefore $\pi_{\mathcal{I}(\mathcal{A})} := \pi_{\mathcal{I}(\mathcal{A}')}$ satisfies \mathcal{A} .
3. *succ*-rule was applied to create \mathcal{A} : Since this rule was applicable neither the \sqcap -rule nor the \sqcup -rule was applicable. We know that there is at least one violated $x : \text{succ}(c)$. Together with all other *succ*-assertion of x we create a *QFBAPA* formula and obtain a solution from a *QFBAPA* solver. Since the solver works correctly we know that in the next step, where we add variables to \mathcal{A} accordingly to ϕ , \mathcal{A} remains clash free. That means, we can obtain an induced interpretation $\mathcal{I}(\mathcal{A})$ by introducing a new element $y^{\mathcal{I}(\mathcal{A})}$ for each $e \in E(\phi)$ (and hence for each newly introduced variable y in \mathcal{A}) and add it to $C^{\mathcal{I}(\mathcal{A})}$ for each $e \in X_C$ and add $(x^{\mathcal{I}(\mathcal{A})}, y^{\mathcal{I}(\mathcal{A})})$ to $r^{\mathcal{I}(\mathcal{A})}$ for each $e \in X_r$. We update the assignment $\pi_{\mathcal{I}(\mathcal{A}')}$ as follows: $\pi_{\mathcal{I}(\mathcal{A})} \cup \{y \mapsto y^{\mathcal{I}(\mathcal{A})} | y \text{ was freshly introduced in this step}\}$.

□

Lemma 2. If $\mathcal{A} := \{x : C\}$ is satisfiable then the Tableau-algorithm terminates without a clash.

Proof. The termination without clashes can be describe as follows:

There exists a chain of of ABoxes $\mathcal{A} = \{x : C\}, \mathcal{A}_1, \dots, \mathcal{A}_n$ such that

- \mathcal{A}_{i+1} can be obtained from \mathcal{A}_i

- every ABox in this chain is satisfiable
- \mathcal{A}_n contains no clash

Since \mathcal{A} is satisfiable the induced interpretation is satisfied by $\pi_{\mathcal{I}(\mathcal{A})} = C^{\mathcal{I}(\mathcal{A})}$ (induction base).

We assume that a already obtained \mathcal{A}_i is satisfiable (induction hypothesis).

We apply a rule to \mathcal{A}_i to obtain \mathcal{A}_{i+1} (induction step):

- we apply the \sqcap -rule: We know that for \mathcal{A}_i we have in induced interpretation such that $\pi_{\mathcal{I}(\mathcal{A}_i)}$ satisfies \mathcal{A}_i . If the \sqcap -rule was applied then $\pi_{\mathcal{I}(\mathcal{A}_i)}$ also satisfies \mathcal{A}_{i+1} .
- we apply the \sqcup -rule: We have to show that either $\mathcal{A}_{i+1} = \mathcal{A}_i \cup \{x : C_1\}$ or $\mathcal{A}_{i+1} = \mathcal{A}_i \cup \{x : C_2\}$ is satisfiable, $x : C_1 \sqcup C_2 \in \mathcal{A}_i$. Again by the induction hypothesis \mathcal{A}_i is satisfiable and hence $\pi_{\mathcal{I}(\mathcal{A}_i)}(x) = (C_1 \sqcup C_2)^{\mathcal{I}(\mathcal{A}_i)}$. So either we choose $\mathcal{A}_{i+1} = \mathcal{A}_i \cup \{x : C_1\}$ and hence $\pi_{\mathcal{I}(\mathcal{A}_i)}(x) = C_1^{\mathcal{I}(\mathcal{A}_{i+1})}$ or we choose $\mathcal{A}_{i+1} = \mathcal{A}_i \cup \{x : C_2\}$ and hence $\pi_{\mathcal{I}(\mathcal{A}_i)}(x) = C_2^{\mathcal{I}(\mathcal{A}_{i+1})}$. In both cases $\pi_{\mathcal{I}(\mathcal{A}_i)}$ satisfies \mathcal{A}_{i+1} .
- we apply the *succ*-rule: If we apply this rule, all boolean rules are in applicable. Because \mathcal{A}_i is satisfiable, we know that a subset of \mathcal{A}_i is also satisfiable and hence also the subset of all *succ*-assertion of x , which is translated to a *QFBAPA* formula. Therefore the *QFBAPA* solver return a solution ϕ . Accordingly to the solution we introduce successors for x and possible add assertion of the form $y : C$, $y : \text{succ}(c)$, $(x, y) : r$, with $y \in \text{Var}(\mathcal{A}_{i+1}) \setminus \text{Var}(\mathcal{A}_i)$. \mathcal{A}_{i+1} remains satisfiable, because we the *QFBAPA* solver returns a satisfiable solution.

Since \mathcal{A} is a finite set and we can not have infinite nested levels, the algorithm can not introduce infinite successors. Together with every Abox in the chain being satisfiable, the algorithm ends with a satisfiable Abox \mathcal{A}_n \square

We know that a subset of a satisfiable ABox \mathcal{A} is also satisfiable because the induced interpretation $\mathcal{I}(\mathcal{A})$, which satisfies \mathcal{A} , satisfies all subsets of \mathcal{A} .

We now prove that the algorithm terminates. For that we define first the depth **and size?** of a concept C :

$$\text{depth}(C) := \begin{cases} 1 + \text{depth}(c) & C = \text{succ}(c) \\ 0 & \text{otherwise} \end{cases}$$

This function is well-define because by Definition 3 our concepts are inductively defined from concept names, hence the *depth*-function always returns an integer.

References

- [1] F. Baader. A New Description Logic with Set Constraints and Cardinality Constraints on role Successors, 2017.

- [2] R. Hoehndorf, P. N. Schofield, and G. V. Gkoutos. The role of ontologies in biological and biomedical research: a functional perspective. *Brief. Bioinform*, page 1069–1080, 2015.
- [3] B. Hollunder and F. Baader. Qualifying Number Restrictions Concept Languages. Technical report, Research Report RR-91-03, 1991.
- [4] S. Tobies. *Complexity Results and Practical Algorithms for Logics in Knowledge Representation*. PhD thesis, Rheinisch-Westfälischen Technischen Hochschule Aachen, 2001.
- [5] S. Tobies and H. Ganzinger. A PSpace Algorithm for Graded Modal Logic. In *In Proc. CADE 1999, Lecture Notes in Artificial Intelligence*, volume 1632, pages 674–674, 01 1999.