Faculty of Computer Science

Institute of Theoretical Computer Science

Chair of Automata Theory

Prof. Dr.-Ing. Franz Baader

Master's Thesis

A Tableau Algorithm for the Numerical Description Logic $\mathcal{ALCSCC}$

Ryny Khy

Matriculation Number: 4751049
born on 30. November 1994 in Landshut

supervised by
Dr.-Ing. Stefan Borgwardt
Filippo De Bortoli M.Sc.

reviewed by
Dr.-Ing. Stefan Borgwardt
Prof. Sebastian Rudolph

November 13, 2020

Declaration of Authorship

Author: Ryny Khy
Matriculation Number: 4751049
Title: A Tableau Algorithm for the Numerical Description Logic $\mathcal{ALCSCC}$

I hereby declare that I have written this final thesis independently and have listed all used sources and aids. I am submitting this thesis for the first time as a piece of assessed academic work. I understand that attempted deceit will result in the failing grade "not sufficient" (5.0).

Place and Date	Author's signiture

Abstract

In the research field of Description Logics (DLs) checking satisfiability of $\mathcal{ALCQ}$ has been investigated thoroughly and is therefore well-known. The DL $\mathcal{ALCSCC}$ extend $\mathcal{ALCQ}$ with constraints over role successor using quantifier-free fragment (QF) of Boolean Algebra (BA) and Presburger Arithmetic (PA). Checking satisfiability of this DL has been proven to be decidable and PSpace-complete. We provide in this work a tableau algorithm for checking satisfiability of $\mathcal{ALCSCC}$ and its correctness proof.

Contents

Chapter 1

Introduction

Traditional databases, where data are stored solely without any connection information, like many people would imagine are often not enough anymore. The reason is that the data are stored without any semantics. However storing data with semantics can provide additional information. For example imagine that we have some data about two objects *"Anna"* and *"Beth"*. In a traditional database, if not explicitly stated, both are not related to each other. However Anna and Beth can have a relation, which also depends on who or what both are. For example both can be human and are assigned to the same class and Anna is a teacher and Beth is a student. By adding solely this information to a traditional database, the information that Anna must teach Beth is not given. One way to apply semantics to data objects is to use *ontologies*. In biological and medical research databases are often based on ontologies [2]. Ontologies (in the computer science field) can be viewed as formal representations of a certain domain of interest. The relations between the entities in the database are formulated in a fragment of first-order logic (FOL). These fragments of FOL are represented as *Description Logic (DL)*, which is a family of knowledge representations system. DLs are mainly built of concepts, which correspond to unary relations in FOL, and relation between the concepts, which correspond to binary relations in FOL. 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 formalized in FOL as $\forall x.Man(x) \vee Woman(x) \rightarrow Human(x)$ and in DL as an *axiom* $Man \sqcup Woman \sqsubseteq Human$, where Man,$Woman$ and $Human$ are concept names. The statement "All Humans who have children are parents" can be formalized in FOL as $\forall x \exists y.Human(x) \wedge hasChild(x, y) \rightarrow Parent(x)$ and in DL as $Human \sqcap \exists hasChild.\top \sqsubseteq Parent$, where $Human$ and $Parent$ are concept names and $hasChild$ is a role name. The second statement can also be formalized with a *quantitative* restriction: $Human \sqcap \geq 1 hasChildren.\top \sqsubseteq Parent$. Each quantified restriction can be transformed into a qualified restriction.
A knowledge base consists of a *TBox*, which contains the axioms (rules), and of an *ABox* which contains assertions about certain elements (objects). One big

research field in DL is the determination of satisfiability of a knowledge base, which is formulated in DL. In [3] a *tableau* algorithm is presented for checking satisfiability for an ABox in the DL $\mathcal{ALCQ}$. This DL allows conjunctions ($\sqcap$), disjunctions ($\sqcup$), negation ($\neg C$) and qualifying number restriction ($\leq\ n\, r\, C$ and $\geq\ n\, r\, C$), where n is a number, r a role name, and C a concept name. A tableau algorithm applies *completion rules* to a given *set* (ABox) to decompose complex concepts and try satisfiying violated assertions. The satisfiability (of concepts) is stated in [3] as a PSPACE-hard problem, with TBox it is EXPTime-hard [1]. In [7] an optimized Tableau-algorithm is presented which shows that it is a PSPACE-problem. The optimization is that instead of keeping n successors to satisfy a restriction $\geq\ n\, r.C$ like in [3], the algorithm instead of saving all successors it only saves an integer which denotes the number of successors and by comparing the numbers detects possible *clashes*.
The expressive DL $\mathcal{ALCSCC}$ [1] extends $\mathcal{ALCQ}$ with *set constraints* and *cardinality constraints* over role successors, which uses the logic of *QFBAPA* [4] (quantifier-free fragment of Boolean Algebra with Presburger Arithmetic). Instead of quantifiers we use set expressions (Boolean Algebra) and numerical constraints (Presburger Arithmetic). For example $Human \sqcap \geq 1\, hasChild.\top \sqsubseteq Parent$ is written in $\mathcal{ALCSCC}$ as $Human \sqcap succ(|hasChild| \geq 1) \sqsubseteq Parent$. This DL is more expressive than $\mathcal{ALCQ}$ because every qualified restriction $\leq\ n\, r.C$ and $\geq\ n\, r.C$ can be written in $\mathcal{ALCSCC}$ as $succ(|r \cap C| \leq 1)$ and $succ(|r \cap C| \geq 1)$ respectively. However a constraint like $succ(|r| = |s|)$ can not be formulated in $\mathcal{ALCQ}$ [1]. In [1] a solution for the satisfiability problem without TBox is presented which has the complexity PSpace: For an ABox a part we guess the value (true or false) of the top-level atoms (concepts), which can already lead to a *false* result, which means the ABox is unsatisfiable. If not, then the ABox is formulated into a QFBAPA formula. We then extend the formula with constraints over the *Venn regions* of the concepts. For the new formula we test whether it returns true or false with a NP satisfiability algorithm for QFBAPA. If the test returns true we are done. If it returns false, we create for every guessed Venn region a concept corresponding to the Venn region. Then the algorithm is applied on this new concept recursively. If it return false, the ABox is unsatisfiable, otherwise satisfiable.
In this work we present a tableau algorithm for $\mathcal{ALCSCC}$. As in previous work for other DLs, we define completion rules which can be applied onto assertions in the ABox. If we can not apply any rules anymore and and the ABox contains a *clash*, means that the ABox contains a contradiction, then the ABox is unsatisfiable, otherwise satisfiable. The main difficulty is that unlike $\mathcal{ALCQ}$, where the bond of number of successor is fixed, in $\mathcal{ALCSCC}$ by adding role successors the cardinalities in the constraint can vary like in $succ(|r| = |s|)$. Hence we need an approach for counting successors and with it calculating the *correct* cardinality, which is necessary to detect satisfied and violated constraints. For this we introduce an *induced interpretation*, which can determine the cardinalities after each rule application. Furthermore, to deal with the numerical arithmetic of $\mathcal{ALCSCC}$ we use a QFBAPA solver, of which we assume it is capable to return all possible solutions. 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 individual names accordingly to our ABox. Since the solver should be capable to return every solution, we have an infinite amount of solutions. Hence we pre compute an upper bound for them and show that we do not loose important information.

Chapter 2

Preliminaries

Before we define the DL $\mathcal{ALCSCC}$ we have to explain first how the language QFBAPA looks like.

2.1 QFBAPA

The logic QFBAPA [4] combines boolean algebra (BA) over a sets of symbols with Presburger arithmetic (PA). A *term* of a boolean algebra over a symbol set T is a conjunction ($\cap$) and/or disjunction ($\cup$) of symbols, which are possibly negated ($s^\neg,\ s \in T$). A term of the Presburger arithmetic are additions of natural numbers. In QFBAPA we construct *set terms* after the boolean algebra and create *cardinality terms* over those set terms with help of the Presburger arithmetic. Since we can construct multiplication with the help of additions, multiplication of cardinality terms is also included. Over the terms we can construct *constraints* in this logic: We can state inclusions of set terms and compare cardinality terms. The formal definition follows.

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

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

  where s, t are set terms
- cardinality terms over T are:

  - every number $n \in \mathbb{N}$
  - $|s|$ if s is a set term
  - if k, l are cardinality terms then also $k + l$ and $n \cdot k$, $n \in \mathbb{N}$

- cardinality constraints over T are:
  - $k = l$ and $k \neq l$
  - $k < l$ and $k \geq l$
  - $k \leq l$ and $k > l$
  - $n \ dvd \ k$ and $n \ \neg dvd \ k$

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

A $QFBABA$ formula ϕ is a disjunction ($\vee$) and conjunction ($\wedge$) of (also possible negated) cardinality constraints, where every set symbol is represented as a set variable.

Since $s \subseteq t$ can be expressed as the cardinality constraint $|s \cap t^{\neg}| \leq 0$ we will not consider any set constraints further in this work. In case we want to express $x : succ(s = t)$, with s, t being set terms, we write instead $x : succ(|s \cap t^{\neg}| \leq 0) \sqcap succ(|s^{\neg} \cap t| \leq 0)$. 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 $l \leq k$.
For an example we have a set of symbols $T = \{l, a, n, e, f\}$ and some constraints like $|l| = 2$, $|l| = |a|$, $|e \cap f^{\neg}| = 0$, $|n \cap f^{\neg}| = 0$. We can connect those constraints to one formula

$$|l| = 2 \wedge |l| = |a| \wedge |e \cap f^{\neg}| = 0 \wedge |n \cap f^{\neg}| = 0 \tag{2.1}$$

To satisfy the formula we have to create a semantic for it such that all cardinality constraint are true (since all constraint are connected with $\wedge$). In our case we would need two elements which are in the semantic of l, hence also two elements in the semantic of a.
The semantics of QFBAPA, called substitutions, is defined as follows:

Definition 2 (Substitutions of QFBAPA)**.** A substituations σ of a over a symbol set T is a mapping that assign

- $\mathcal{U}$ to a finite set $\sigma(\mathcal{U})$
- every symbol a in T to $\sigma(a) \subseteq \sigma(\mathcal{U})$
- $\emptyset$ to $\sigma(\emptyset) = \emptyset$
- $\sigma(s \cap t) := \sigma(s) \cap \sigma(t)$, $\sigma(s \cup t) := \sigma(s) \cup \sigma(t)$
- $\sigma(s^{\neg}) := \sigma(\mathcal{U}) \backslash \sigma(s)$
- $\sigma(|s|) := |\sigma(s)|$

- $\sigma(k+l) := \sigma(k) + \sigma(l)$, $\sigma(n \cdot k) := n \cdot \sigma(k)$

Given cardinality terms k, l we say that σ satisfies

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

The substitution σ is a solution of a QFBAPA formula ϕ if the formula with σ can be evaluated to $\top$. A QFBAPA formula is satisfiable if it has a solution σ and is unsatisfiable otherwise.
A solution to (2.1) can the following: Let $\sigma(\mathcal{U}) = \{$leg1, leg 2, arm1, arm 2, nose, ear1, ear2$\}$ and:

- $\sigma(l) = \{$leg1, leg2$\}$
- $\sigma(a) = \{$arm1, arm2$\}$
- $\sigma(n) = \{$nose$\}$
- $\sigma(e) = \{$ear1, ear 2$\}$
- $\sigma(f) = \{$nose, ear1, ear2$\}$

The interpretation satisfies the formula because $\sigma(|l|) = 2 = \sigma(|k|)$ and $\sigma(|n \cap f^{\neg}|) = 0$ and $\sigma(|e \cap f^{\neg}|) = 0$.
Now we can interpret our formula as

- we have 2 legs
- we have as many legs as arms
- nose and two ears are both in the same set hence they belong to a common body part (face)

2.2 $\mathcal{ALCSCC}$

Next we define the description logic $\mathcal{ALCSCC}$ [1]. 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 ($\mathcal{ALCSCC}$)**.** $\mathcal{ALCSCC}$ concepts over $\mathbf{C}$ and $\mathbf{R}$ are defined inductively:

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

An $\mathcal{ALCSCC}$ ABox $\mathcal{A}$ is a finite set of assertions of the form $x : C$ and $(x, y) : r$, where C is a $\mathcal{ALCSCC}$ concept, $r \in \mathbf{R}$ and x, y are individual names. The set $I(\mathcal{A})$ is the set of individual names occurring in $\mathcal{A}$.
Regarding our QFBAPA example 2.1 we can now construct an ABox of specific individual names, in which we describe their bodies. Let $\mathbf{C} = \{Legs, Arms, Female\}$ and $\mathbf{R} = \{bodyParts\}$. A possible ABox, which states that an individual name *Anna* has two legs and two arms and is female, is

$$\{Anna : succ(|Legs \cap bodyParts| = 2) \sqcap succ(|Legs| = |Arms|) \sqcap Female\} \quad (2.2)$$

Similar to the substitutions for QFBAPA we define now the semantics, called interpretations, for $\mathcal{ALCSCC}$

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

- each individual name $x \in I(\mathcal{A})$ 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 $(x, y) \in r^{\mathcal{I}}$ i.e. it contains all r-successors of x.
For compound concepts the mapping $\cdot^{\mathcal{I}}$ is extended inductively as follows

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

The mapping $\cdot^{\mathcal{I}_x}$ is a QFBAPA substitution that 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 $r^{\mathcal{I}_x} := r^{\mathcal{I}}(x)$.
$\mathcal{I}$ is a model of $\mathcal{A}$ iff

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

We can define a model $\mathcal{I}$ of the ABox defined in (2.2) by setting $\Delta^{\mathcal{I}} = \{Anna^{\mathcal{I}}, Leg1^{\mathcal{I}}, Leg2^{\mathcal{I}}, Arm1^{\mathcal{I}}, Arm2^{\mathcal{I}}\}$ and

- $Female^{\mathcal{I}} = \{Anna^{\mathcal{I}}\}$
- $Leg^{\mathcal{I}} = \{Leg1^{\mathcal{I}}, Leg2^{\mathcal{I}}\}$

- $Arm^{\mathcal{I}} = \{Arm1^{\mathcal{I}}, Arm2^{\mathcal{I}}\}$
- $bodyPart^{\mathcal{I}} = \{(Anna, Leg1), (Anna, Leg2), (Anna, Arm1), (Anna, Arm2)\}$

By mapping $Anna$ to $Anna^{\mathcal{I}}$, $Leg1$ to $Leg1^{\mathcal{I}}$ and so on we see that this interpretation satisfies the ABox: $Anna^{\mathcal{I}}$ is indeed in $succ(|Legs \cap bodyParts| = 2)^{\mathcal{I}}$ because $Leg1^{\mathcal{I}}, Leg2^{\mathcal{I}} \in bodyPart^{\mathcal{I}_{Anna}} \cap Leg^{\mathcal{I}_{Anna}}$. Analogously for the second $succ$-assertion.
Lastly we define the *negated normal form* (NNF) of $\mathcal{ALCSCC}$. By transforming all concept into NNF we avoid nested negation e.g. $\neg(\neg(\neg(A \cup B)))$ which helps to formulate the rules for the Tableau algorithm.

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 anymore.

- $\neg\top \to \bot$
- $\neg \bot \to \top$
- $\neg\neg C \to C$
- $\neg(C \sqcap D) \to \neg C \sqcup \neg D$
- $\neg(C \sqcup D) \to \neg C \sqcap \neg D$
- $C^{\neg} \to \neg C$
- $\neg succ(c) \to succ(\neg c)$
- $\neg(k \leq l) \to l \leq k$
- $\neg(n\ dvd\ k) \to n\ \neg dvd\ k$
- $\neg(n\ \neg dvd\ k) \to n\ dvd\ k$
- $(s \cap t)^{\neg} \to s^{\neg} \cup t^{\neg}$
- $(s \cup t)^{\neg} \to s^{\neg} \cap t^{\neg}$
- $(s^{\neg})^{\neg} \to s$

The rule $C^{\neg} \to \neg C$ 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 σ for a concept C based on QFBAPA we have $\sigma(C^{\neg}) = \sigma(\mathcal{U}) \backslash \sigma(C)$ and for every interpretation $\mathcal{I}$ based on $\mathcal{ALCSCC}$ we have $(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \backslash C^{\mathcal{I}}$. Since $\sigma(\mathcal{U}) \subseteq \Delta^{\mathcal{I}}$ we can conclude that every element in $\sigma(C^{\neg})$ is also in $(\neg C)^{\mathcal{I}}$.
The first five rules on the left hand side can be applied in linear time [3],[6]. The first four rules on the right hand side and the rules $C^{\neg} \to \neg C$ and $\neg succ(c) \to succ(\neg c)$ can also be applied in linear time since we only shift the negation sign. The rule $(s^{\neg})^{\neg} \to s$ works similarly to $\neg\neg C \to C$ and the rules $(s \cap t)^{\neg} \to s^{\neg} \cup t^{\neg}$ and $(s \cup t)^{\neg} \to s^{\neg} \cap t^{\neg}$ works similarly to $\neg(C \sqcap D) \to \neg C \sqcup \neg D$ and $\neg(C \sqcap D) \to \neg C \sqcup \neg D$. Therefore all of them can also be applied in linear time. Regarding the normal form we also replace in every $succ(c)$ concept every disjunction and conjunction in form as $\sqcap$ and $\sqcup$ with $\cap$ and $\cup$. We can do this because for an arbitrary interpretation $\mathcal{I}$ for each $x, y \in \Delta^{\mathcal{I}}$ we have $y \in (C \sqcap D)^{\mathcal{I}_x}$ iff $y \in (C \cap D)^{\mathcal{I}_x}$:

$$
\begin{aligned}
& y \in (C \sqcap D)^{\mathcal{I}_x} && \leftrightarrow \\
& y \in (C \sqcap D)^{\mathcal{I}} \cap \mathcal{U}^{\mathcal{I}_x} && \leftrightarrow \\
& y \in C^{\mathcal{I}} \cap D^{\mathcal{I}} \cap \mathcal{U}^{\mathcal{I}_x} && \leftrightarrow \\
& y \in (C^{\mathcal{I}} \cap \mathcal{U}^{\mathcal{I}_x}) \cap (D^{\mathcal{I}} \cap \mathcal{U}^{\mathcal{I}_x}) && \leftrightarrow \\
& y \in C^{\mathcal{I}_x} \cap D^{\mathcal{I}_x} && \leftrightarrow \\
& y \in (C \cap D)^{\mathcal{I}_x}
\end{aligned}
$$

Chapter 3

Tableau for $\mathcal{ALCSCC}$

A Tableau-algorithm consists of completion rules to decide satisfiability an ABox. The rules are applied exhaustively on the set until none is applicable anymore. Originally one major characteristic of a traditional tableau algorithm is that it does not matter in which order the rules are applied. However for this DL we have to give the rules priorities. Another characteristic of a tableau algorithm, which is kept for this DL, is that it works non-deterministically: In case we have disjunctions we can choose between the concepts in this disjunction. If a choice ends in a *clash* then we back track to the point where we had to choose and take the other choice instead. If all choices end 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 by applying the rules on the ABox $\mathcal{A} = \{x : C\}$.
To define *clash* we first introduce *induced interpretation* with which we count successors of any individual name after any rule application and hereby detect violated assertions.

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

- for each individual name $x \in I(\mathcal{A})$ we introduce $x^{\mathcal{I}(\mathcal{A})}$ 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 $r^{\mathcal{I}(\mathcal{A})}$

Definition 7 (Violated assertion)**.** Let $\mathcal{A}$ be an ABox, x be an individual name, k be a cardinality term and $n \in \mathbb{N}$. An assertion $x : succ(c)$ is *violated* if $x^{\mathcal{I}(\mathcal{A})} \notin succ(c)^{\mathcal{I}(\mathcal{A})}$.

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

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

- $\{x :\bot\} \subseteq \mathcal{A}$ or

- $\{x : C, x : \neg C\} \subseteq \mathcal{A}$ or
- $\{(x, y) : r, (x, y) : \neg r\} \subseteq \mathcal{A}$ or
- $x : succ(c) \in \mathcal{A}$ 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 a QFBAPA solver whenever we want to add successors for an individual name x. For that we collect all *succ*-assertions regarding x first and then transform them into a QFBAPA formula for the upper *nested level*, which means we do not consider the successors of x's successors yet. We assume that the ABox is already in NNF. As example we look at

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

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

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:

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

We replace (possibly compound) concepts and role names with set variables, for which a solver can assign elements to them. The last bullet point is important because sometimes it is not explicitly stated what kind of successor an individual name has. That means that every successor is connected to its predecessor by at least one role name.
For our example we have five set variables: X_A, X_B, X_r, $X_{succ(|A| \leq |B \cap r|)}$ and $\mathcal{U}$. The QFBAPA formula for Example 1 is

$$\phi = 1 \leq |X_{succ(|A| \leq |B \cap r|)}| \wedge |X_A| \leq |X_B| \wedge \mathcal{U} = X_r \tag{3.1}$$

We only replace concepts on one *nested level*. In the example the first assertion says that x must have at least one successor y which has at least as many successors in $B \cap r$ as in A. The concept $succ(|A| \leq |B \cap r|)$ is on a different *nested level* than $succ(|A| \leq |B|)$. Also x has at least as many successors in B as in A. By the last conjunction we give the information that every successor must be an r-successor. We can now let the solver gives us a possible solution, if there is one.

3.2 Solution of a formula

For our algorithm we assume that we have a solver which can return all possible solutions. However there can be infinitely many solutions. Actually Example 1 can have infinitely many solutions: We can always increase the amount of successors in B as long as we have fewer successors in A. However having that means that the tableau algorithm works on an infinite space and/or does not terminate. Therefore we want to only consider solutions inside some *upper bounds*. For an *Integer Linear Programming* (ILP) problem, which is described as a system of linear equalities, possible upper bounds are already investigated (like in [5]). Therefore we want to transform our formula into a linear system of equalities of the form $Ax = b$, where A and b describe our cardinality constraints and x is the solution i.e. denotes the numbers of elements we have to assign to set variables to satisfy the formula:
First, we notice that every inequality in a QFBAPA formula can be rewritten as $n_1 \cdot |X_1| \pm \cdots \pm n_i \cdot |X_i| \lesseqgtr I$, $\lesseqgtr \in \{\leq, \geq, =\}$, where $n_1, \ldots, n_i, I \in \mathbb{Z}$ are a constants. The numbers $n_1, \ldots, n_i$ are called *pre-factors*. Let $c = succ(|A| \leq |B \cap r|)$. We arrange ϕ in (3.1) into

$$\phi' = |X_c| \geq 1 \wedge |X_A| - |X_B| \leq 0 \wedge |\mathcal{U} \cap X_r^{\neg}| = 0 \wedge |\mathcal{U}^{\neg} \cap X_r| = 0$$

Then we change each inequalities into equalities by adding two slack variables I_1 and I_2:

$$\phi'' = |X_c| - I_1 = 1 \wedge |X_A| - |X_B| + I_2 = 0 \wedge |\mathcal{U} \cap X_r^{\neg}| = 0 \wedge |\mathcal{U}^{\neg} \cap X_r| = 0$$

Right now it is not clear whether the set variables are overlapping or not. Therefore we consider *Venn regions*, which are of the form $X_1^i \cap \cdots \cap X_k^i$. The subscript i denotes either 0 or 1. X_1^0 denotes $X_1^{\neg}$ and X_1^1 denotes X_1. Therefore we construct A and b such that instead of assigning elements to set variables we assign them to the Venn regions. Since we have five set variables, we have $2^5 = 32$ Venn regions and therefore 32 vectors denoting the Venn regions. We also have four equations and two more vectors for the slack variables. The matrix A therefore has four rows and 34 columns with a_{ij}, $1 \leq i \leq 4$ and $1 \leq j \leq 34$, denoting the sum of the pre-factors of the set variables, in which the j-th Venn region is included in the i-th equation. We already see that the number of Venn regions grows exponentially with the number of set variables. In [1] it is stated that there exists a number N, which is polynomial in the size of ϕ, such that at most N Venn regions are not empty if there exists a solution.

Lemma 1 (Lemma 3 from [1])**.** For every QFBAPA formula ϕ a number N, which is polynomial in the size of ϕ, can be computed in polynomial time such that for every solution σ of ϕ there exists a solution σ' of ϕ such that

- $|\{v | v \text{ is a Venn region and } \sigma'(v) \neq \emptyset\}| \leq N$
- $\{v | v \text{ is a Venn region and } \sigma'(v) \neq \emptyset\} \subseteq \{v | v \text{ is a Venn region and } \sigma'(v) \neq \emptyset\}$

Hence we guess (in non-deterministic polynomial time) a number N of Venn regions, which are non-empty. In Example 1 we can already guess that any Venn region within $X_r^\neg$ or $\mathcal{U}^\neg$ must be empty, because every element must be in $\mathcal{U}$ and since $\mathcal{U} = X_r$ they must be all in X_r. Hence we can drop 24 Venn regions and create a system of equations with A being a 4×10 matrix. Let the vectors for the Venn regions be in the following order:

- $v_1 = X_A \cap X_B \cap X_c \cap X_r \cap \mathcal{U} \cap I_1^\neg \cap I_2^\neg$
- $v_2 = X_A \cap X_B \cap X_c^\neg \cap X_r \cap \mathcal{U} \cap I_1^\neg \cap I_2^\neg$
- $v_3 = X_A \cap X_B^\neg \cap X_c \cap X_r \cap \mathcal{U} \cap I_1^\neg \cap I_2^\neg$
- $v_4 = X_A \cap X_B^\neg \cap X_c^\neg \cap X_r \cap \mathcal{U} \cap I_1^\neg \cap I_2^\neg$
- $v_5 = X_A^\neg \cap X_B \cap X_c \cap X_r \cap \mathcal{U} \cap I_1^\neg \cap I_2^\neg$
- $v_6 = X_A^\neg \cap X_B \cap X_c^\neg \cap X_r \cap \mathcal{U} \cap I_1^\neg \cap I_2^\neg$
- $v_7 = X_A^\neg \cap X_B^\neg \cap X_c \cap X_r \cap \mathcal{U} \cap I_1^\neg \cap I_2^\neg$
- $v_8 = X_A^\neg \cap X_B^\neg \cap X_c^\neg \cap X_r \cap \mathcal{U} \cap I_1^\neg \cap I_2^\neg$

The last two vectors denote the slack variables

- $v_9 = X_A^\neg \cap X_B^\neg \cap X_c^\neg \cap X_r^\neg \cap \mathcal{U}^\neg \cap I_1 \cap I_2^\neg$
- $v_{10} = X_A^\neg \cap X_B^\neg \cap X_c^\neg \cap X_r^\neg \cap \mathcal{U}^\neg \cap I_1^\neg \cap I_2$

We create now the linear system of equation:

$$\begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & -1 & -1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \\ x_{10} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \end{pmatrix}$$

Note that $a_{2,1}$ and $a_{2,2}$ are 0 because the pre-factors of $|X_A|$ and $|X_B|$ in the second equation are 1 and -1 and the Venn regions v_1 and v_2 are included both in X_A and X_B. If $x_i = 0$, then the Venn region v_i is empty. The last two rows of A are lines of zeros because we omit the Venn regions, in which $\mathcal{U} \cap X_r^\neg$ and $\mathcal{U}^\neg \cap X_r$ are included. However we do not loose information because the information is that those Venn regions must be empty.
Going back to the upper bound problem: The following result from [4] can be used to establish an upper bound for the solution of the *ILP*:

Theorem 1 (Theorem 1 from [5])**.** Let $A \in \mathbb{Z}^m \times \mathbb{Z}^n$ be a matrix and $b \in \mathbb{Z}^m$ a vector. If $x \in \mathbb{N}^n$ is a solution of $Ax = b$, then there exists a solution x' such that all entries are integers between 0 and $n \cdot (m \cdot max_{i,j}\{|a_{ij}|, |b_i|\})^{2 \cdot m+1}$.

We take a look now in the proof of this theorem to understand how the solution is decreased. We distinguish between two cases. Let $M = m \cdot max_{i,j}\{|a_{ij}|\}^m$, $F = \{i | x_i > M\}$ and v_i be the i-th column of A

- If there exist integers α_i, for all $i \in F$, such that $\sum_{i \in F} \alpha_i \cdot v_i = 0$ and $\exists i : \alpha_i > 0$ then $x' = x - d$, $d_j = \alpha_j$ if $j \in M$ else $d_j = 0$, $1 \leq j \leq n$.
- Else: There must be a vector $h \in \{0, \pm 1, \pm 2, \cdots \pm M\}^m$ such that $h^T v_i \geq 1, i \in F$. We premultiply A and b with h^T and show that x is already in the bound:

$$h^T Ax = h^T b$$

Therefore we are able to set an upper bound a priori for the solutions the QFBAPA solver returns. Furthermore we stay in PSpace after Lemma 1 because the calculation of the upper bound use $\leq N$ Venn regions.
In our example the upper bound for all x_i is $10 \cdot (2 \cdot max\{|1|, |-1|\})^{2 \cdot 2+1} = 320$, which means that we can discard any solution in which a Venn region has more than 320 elements.

3.3 The Tableau Algorithm

Finally we can present the Tableau-algorithm for an ABox in $\mathcal{ALCSCC}$. Before we handle the numerical arithmetic of $\mathcal{ALCSCC}$ we want to decompose compound concepts first. Hence we divide the algorithm in two parts: a boolean part, where the decomposition of compound concepts takes place, and a numerical part, where a part of the ABox is transformed into a QFBAPA formula. Then we calculate an upper bound and let a solver return a possible solution within this bound, in case the ABox is satisfiable. The boolean part has a higher priority than the numerical part.

Definition 9 (Tableau for $\mathcal{ALCSCC}$)**.** The completion rules for a $\mathcal{ALCSCC}$ ABox $\mathcal{A}$ in NNF are the following.
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 an individual name x at least one violated assertion of the form $x : succ(c)$ and no boolean rule is applicable:

- gather all assertions of the form $x : succ(c)$ into a set $\mathcal{S}$
- transform $\mathcal{S}$ into a QFBAPA formula ϕ as in Section 3.1
- calculate the upper bound as in Theorem 1

If the QFBAPA solver returns *unsatisfiable*, then $\mathcal{A} := \mathcal{A} \cup \{x : \bot\}$
If the QFBAPA solver returns *satisfiable*, then select one solution σ with in the upper bound. For each $e \in \sigma(\mathcal{U})$, we introduce a new individual name y_e and

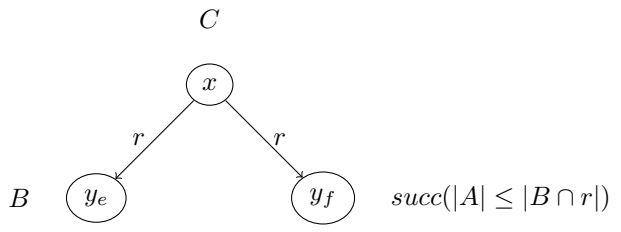
- if $e \in X_C$ we set $\mathcal{A} := \mathcal{A} \cup \{y_e : C\}$
- if $e \in X_{succ(c)}$ we have $\mathcal{A} = \mathcal{A} \cup \{y_e : succ(c)\}$
- if $e \in X_r$, $r \in \mathbf{R}$, we set $\mathcal{A} := \mathcal{A} \cup \{(x, y_e) : r\}$
- if $e \notin X_C$ we set $\mathcal{A} := \mathcal{A} \cup \{y_e : \neg C\}$
- if $e \notin X_{succ(c)}$ we set $\mathcal{A} := \mathcal{A} \cup \{y_e : NNF(\neg succ(c))\}$
- if $e \notin X_r$, $r \in \mathbf{R}$, we have $\mathcal{A} := \mathcal{A} \cup \{(x, y_e) : \neg r\}$

A *complete* ABox is an ABox to which no more rules of the Tableau-algorithm are applicable.
We present a possible run of this algorithm over the following ABox:

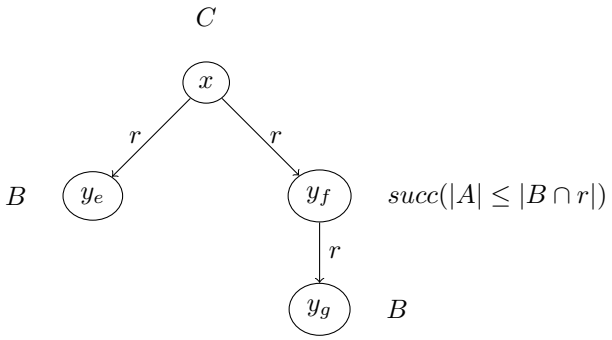
Example 2.

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

We are able to apply to apply the $\sqcap$-rule and hence we apply them first and derive the ABox in Example 1. Then neither the $\sqcap$- nor the $\sqcup$-rule is applicable. Therefore 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 in (3.1). The upper bound for that formula is 320 (see Section 3.2). If the formula have been unsatisfiable, we would add $x : \bot$ to our ABox. However since the formula is satisfiable we obtain a solution from the solver. We see that the formula is satisfiable with $X_{succ(|A| \leq |B \cap r|)} = \{f\}, X_A = \{\}, X_B = \{e\}$ and $X_r = \{e, f\}$. Since we have $\mathcal{U} = \{X_r\}$ every element must be in X_r i.e. every successor is a r-successor. Obviously in this solution every Venn-region has less element than 320. We then introduce for e and f two individual names y_e and y_f and the assertion $y_e : X_B$, $(x, y_e) : r$, $y_f : succ(|A| \leq |B \cap r|)$ and $(x, y_f) : r$ to $\mathcal{A}$.



Then for y_f we have to apply the *successor*-rule again. Note that no boolean rule is applicable. Since $y_f : succ(|A| \leq |B \cap r|)$ is the only *succ* assertion we only have to add a successor of the concept $B \cap r$. We assume that the QFBAPA solver returns a solution $X_B = X_r = \{g\}$ and hereby introduce an individual name y_g and add $y_g : B$ and $(x, y_g) : r$ to $\mathcal{A}$.



In both application of the *successor*-rule we assumed that the solve returns a solution with minimal elements. However note that bigger solutions leads to a clash-free complete ABox. The reason is that if $\mathcal{A}$ is satisfiable, then also every subset of $\mathcal{A}$, hence also $\mathcal{S}$. Because we assume that the QFBAPA solver runs correct it will always return a solution. Also note that the algorithm considers first all assertion of one individual name before consider the next one. In this procedure we do not add any assertions of individual names which are not freshly introduce. Hence if we applied all possible rules on the individual name x and we do not have a clash, then all assertions of x remains satisfied until the tableau algorithm ends. In the next chapter we provide a formal proof of the its correctness.

Chapter 4

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 ABox $\mathcal{A}$, then $\mathcal{A}$ is satisfiable.
- If $\mathcal{A}$ is satisfiable, then the Tableau-algorithm terminates without a clash.

In all proofs we assume that the QFBAPA solver is correct. First we prove that the algorithm always terminates.

4.1 Termination

We define a *derived* ABox as an ABox $\mathcal{A}_2$ after a finite number of rule applications on an ABox $\mathcal{A}_1$. For the termination proof we map every ABox $\mathcal{A}$ to an element $\Psi(\mathcal{A})$ of a set Q. Let $\prec$ be a well-founded strict partial ordering. Since $\prec$ is well-founded we can not have a infinite decreasing chain. Therefore if we show that for every ABox $\mathcal{A}'$, which is derivable from an ABox $\mathcal{A}$, we have $\Psi(\mathcal{A}') \prec \Psi(\mathcal{A})$, then the termination of the algorithm can be concluded.
The elements of Q are multisets of triples. Each triple consists of a multiset and an integer. A multiset M is smaller than a multiset M' if we can obtain M from M' by replacing at least one element of M' with a set of elements, which are smaller. For example the multiset $M = \{2, 2, 2, 1, 5\}$ is smaller then $M' = \{2, 3, 5\}$ because the second entry $\{3\}$ of M' can be replaced by $\{2, 2, 1\}$. In our definition of Q $\Psi(\mathcal{A}')$ is smaller than $\Psi(\mathcal{A})$ if we can replace at least one triple of $\Psi(\mathcal{A})$ with smaller triple to obtain $\Psi(\mathcal{A}')$. A triple (x_1, x_2, x_3) is smaller than a triple (y_1, y_2, y_3) if for the first $i \in \{1, 2, 3\}$ we have $x_i \neq y_i$, then $x_i < y_i$ (lexicographical ordering from left to right). Otherwise (y_1, y_2, y_3) is smaller or equal.
Next we define the size of a concept C inductively:

- $size(A) = 1$ if $A \in \mathbf{C}$

- $size(\neg C) = size(C) + 1$
- $size(C \sqcap D) = size(C \sqcup D) = size(C) + size(D) + 1$
- $size(succ(c)) = \begin{cases} 1 + max\{size(k), size(l)\} & c = k \leq l \\ 1 + size(l) & c = n \, dvd \, l \end{cases}$

The size of a cardinality terms k is also defined inductively:

- $size(n) = 1$ if n is an integer
- $size(r) = size(C) = 1$ if $r \in \mathbf{R}$, $C \in \mathbf{C}$
- $size(|k|) = size(k)$
- $size(k^{\neg}) = size(k) + 1$
- $size(k \cap l) = size(k \cup l) = size(k) + size(l) + 1$

Definition 10. Let $\mathcal{A}$ be a derived ABox. The multiset $\Psi(\mathcal{A})$ consists of triples. Each triple $\psi_{\mathcal{A}}(x)$ represent one individual name x:

- The first component is the integer $max\{size(C)|x : C \in \mathcal{A}\}$.
- The second component is a multiset which contains for each assertion $x : C \sqcap D \in \mathcal{A}$ for which the $\sqcap$-rule is applicable the integer $size(C \sqcap D)$. Respectively for $x : C \sqcup D$.
- The third component is the number of $x : succ(c) \in \mathcal{A}$ for which the *successor*-rule is applicable.

For the ABox $\mathcal{A}$ in Example 2 we have the following multiset:

$$\Psi(\mathcal{A}) = \{\psi_{\mathcal{A}}(x)\} = \{(7, \{7, 7\}, 0)\}$$

After the decomposing we got the ABox $\mathcal{A}^1$, which is stated in Example 1.

$$\Psi(\mathcal{A}^1) = \{\psi_{\mathcal{A}^1}(x)\} = \{(7, \{\}, 2)\}$$

We can see that the second entry of $\psi_{\mathcal{A}'}(x)$ decreases because we do not have any conjunction anymore. Hereby he increment of the third entry does not matter.
Then the *successor*-rule is applied and we add two new individual names y_e and y_f to obtain $\mathcal{A}^2$, hence a new multiset has to be added.

$$\Psi(\mathcal{A}^2) = \{\psi_{\mathcal{A}^2}(x), \psi_{\mathcal{A}^2}(y_e), \psi_{\mathcal{A}^2}(y_f)\} = \{(7, \{\}, 0), (1, \{\}, 0), (3, \{\}, 1)\}$$

We see that $\psi_{\mathcal{A}^2}(x)$, $\psi_{\mathcal{A}^2}(y_e)$ and $\psi_{\mathcal{A}^2}(y_f)$ are smaller than $\psi_{\mathcal{A}^1}(x)$ hence $\Psi(\mathcal{A}^2) \prec \Psi(\mathcal{A}^1)$.
We then apply the last *successor*-rule to y_f and gain

$$\Psi(\mathcal{A}^3) = \{\psi_{\mathcal{A}^3}(x), \psi_{\mathcal{A}^3}(y_e), \psi_{\mathcal{A}^3}(y_f), \psi_{\mathcal{A}^3}(y_g)\} = \{(7, \{\}, 0), (1, \{\}, 0), (3, \{\}, 0), (2, \{\}, 0)\}$$

The newly introduced triple is smaller then $\psi_{\mathcal{A}^2}(y_f)$ and hence we have $\Psi(\mathcal{A}^3) \prec \Psi(\mathcal{A}^2)$.
We are now able to finally prove the termination of the algorithm.

Lemma 2. For any ABox $\mathcal{A} = \{x : C\}$ the Tableau-algorithm terminates

Proof. We show that if $\mathcal{A}'$ is obtainable from $\mathcal{A}$ by a rule from Definition 3.3, then we have $\Psi(\mathcal{A}') \prec \Psi(\mathcal{A})$.

- $\mathcal{A}'$ is obtained from $\mathcal{A}$ by applying the $\sqcap$-rule: The first component remains unchanged because $size(C) \leq size(C \sqcap D)$ and $size(D) \leq size(C \sqcap D)$. The second component becomes smaller because the integer for $size(C \sqcap D)$ is removed (because we cannot apply this rule anymore after one application). In case C and/or D happens to be a disjunction or conjunction the multiset still becomes smaller because $size(C) < size(C) + size(D) + 1 = size(C \sqcap D)$ (respectively for $size(D)$). Hence the triple $\psi_{\mathcal{A}'}(x)$ always becomes smaller.

- $\mathcal{A}'$ is obtained from $\mathcal{A}$ by applying the $\sqcup$-rule: similar to above

- $\mathcal{A}'$ is obtained from $\mathcal{A}$ by applying the *successor*-rule: The first component remains unchanged because we do not add any assertion for x. Because we are able to apply this rule, both $\sqcap$-rule and $\sqcup$-rule are not applicable on $\mathcal{A}$. Hence the second component of $\psi_{\mathcal{A}}(x)$ remains unchanged. In this rule we gather all *succ*-assertion of x and convert them to a QFBAPA formula. If a solver returns unsatisfiable for the formula we have add $x : \bot$ which means we end with a clash. If the solver returns a solution, we add successors according to the solution. Hence we can not apply the *successor*-rule on all *succ*-assertions of x anymore. Therefore $\psi_{\mathcal{A}'}(x)$ is smaller than $\psi_{\mathcal{A}}(x)$.
  For every freshly introduced individual name y we have to add a triple $\psi_{\mathcal{A}'}(y)$ to Q. Since y is a successor the first component of $\psi_{\mathcal{A}'}(y)$ is always smaller than the first component of $\psi_{\mathcal{A}'}(x)$: We know that $max\{size(C)|y : C \in \mathcal{A}'\}$ is always smaller then $max\{size(C)|x : C \in \mathcal{A}'\}$ by the definition of $size(C)$: For each $y : C \in \mathcal{A}'$ we know that C must be part of a cardinality constraint c such that $x : succ(c) \in \mathcal{A}$ and therefore $size(succ(c)) > size(C)$. For any other individual name y_x, such that $y \neq x$ and y was not freshly introduced, the triple $\psi_{\mathcal{A}}(y)$ remains unchanged.

Hence in all three cases we can obtain $\Psi(\mathcal{A}')$ from $\Psi(\mathcal{A})$ by replacing $\psi_{\mathcal{A}}(x)$ with the smaller triple $\psi_{\mathcal{A}'}(x)$. For any newly introduced individual names we showed that the corresponding triples are always smaller than $\psi_{\mathcal{A}'}(x)$. Therefore $\Psi(\mathcal{A}') \prec \Psi(\mathcal{A})$.
Because we work with non-negative integers, the strict partial ordering is indeed well-founded. □

4.2 Soundness and Completeness

After we proved that the algorithm terminates, we continue with the correctness of the algorithm e.g. the algorithm terminates with a clash-free ABox iff the ABox is satisfiable.

Lemma 3. If the Tableau-algorithm is applied on an ABox $\mathcal{A} = \{x : C\}$ and terminates without a clash, then $\mathcal{A}$ is satisfiable

Proof. Let $\mathcal{A}'$ be the result after the algorithm terminated. Since we do not remove any assertion during the algorithm we have $\mathcal{A} \subseteq \mathcal{A}'$. Hence if an interpretation $\mathcal{I}$ satisfies $\mathcal{A}'$ then it also satisfies $\mathcal{A}$. Let $\mathcal{I}(\mathcal{A}')$ be the induced interpretation of $\mathcal{A}'$. We show that $\mathcal{I}(\mathcal{A}')$ indeed satisfies $\mathcal{A}'$ by induction over concepts:
For each concept name $C \in \mathbf{C}$ such that $x : C \in \mathcal{A}'$, we have $x^{\mathcal{I}(\mathcal{A}')} \in C^{\mathcal{I}(\mathcal{A}')}$ by the definition of induced interpretation. (induction base)
We consider $x : C$ where C is a compound concept (induction step):

- $C = \neg D$: Since $\mathcal{A}'$ does not contain a clash, $x : C \in A$ implies $x : D \notin A$. D must be a concept name, because $\mathcal{A}'$ is in NNF. Therefore by definition of induced interpretation and $x : D \notin A$ we have $x^{\mathcal{I}(\mathcal{A}')} \notin D^{\mathcal{I}(\mathcal{A}')}$ which implies $x^{\mathcal{I}(\mathcal{A}')} \in \Delta^{\mathcal{I}(\mathcal{A}')} \backslash D^{\mathcal{I}(\mathcal{A}')} = C^{\mathcal{I}(\mathcal{A}')}$.

- $C = D \sqcap E$: Since the algorithm terminated, the $\sqcap$-rule is not applicable anymore. That means that there is an individual name x, such that $\{x : D, x : E\} \subseteq \mathcal{A}'$. By the induction hypothesis we have $x^{\mathcal{I}(\mathcal{A}')} \in D^{\mathcal{I}(\mathcal{A}')}$ and $x^{\mathcal{I}(\mathcal{A}')} \in E^{\mathcal{I}(\mathcal{A}')}$. Therefore $x^{\mathcal{I}(\mathcal{A}')} \in D^{\mathcal{I}(\mathcal{A}')} \cap E^{\mathcal{I}(\mathcal{A}')} = C^{\mathcal{I}(\mathcal{A}')}$.

- $C = D \sqcup E$: Since the algorithm terminated, the $\sqcup$-rule is not applicable anymore. That means that there is an individual name x, such that $\{x : D, x : E\} \cap \mathcal{A}' \neq \emptyset$. By the induction hypothesis we have $x^{\mathcal{I}(\mathcal{A}')} \in D^{\mathcal{I}(\mathcal{A}')}$ or $x^{\mathcal{I}(\mathcal{A}')} \in E^{\mathcal{I}(\mathcal{A}')}$. Therefore $x^{\mathcal{I}(\mathcal{A}')} \in D^{\mathcal{I}(\mathcal{A}')} \cup E^{\mathcal{I}(\mathcal{A}')} = C^{\mathcal{I}(\mathcal{A}')}$.

- $C = succ(c)$: Since $\mathcal{A}'$ does not contain a clash, the QFBAPA solver must have returned a solution. If the solution is empty, then no individual names are needed to be introduced to satisfy $x : C$ and we have $x^{\mathcal{I}(\mathcal{A}')} \in C^{\mathcal{I}(\mathcal{A}')}$. If the solution is not empty, then the induced interpretation is updated by introducing a new element $y_e^{\mathcal{I}(\mathcal{A}')}$ for each $e \in \sigma(\mathcal{U})$ and hence also for each freshly introduced individual name y_e. For each $e \in X_C$ we have $y : C \in \mathcal{A}'$. By the induction hypothesis $y_e^{\mathcal{I}(\mathcal{A}')}$ must be in $C^{\mathcal{I}(\mathcal{A}')}$. For each $e \in X_r$ we have $(x, y) : r \in \mathcal{A}'$. Since we also added $(x, y) : r$ to $\mathcal{A}'$, we must have $(x^{\mathcal{I}(\mathcal{A}')}, y^{\mathcal{I}(\mathcal{A}')}) \in r^{\mathcal{I}(\mathcal{A}')}$. Lastly for each $e \in X_{succ(d)}$ we have $y : succ(d) \in \mathcal{A}'$. Again by the induction hypothesis $y^{\mathcal{I}(\mathcal{A}')} \in succ(d)^{\mathcal{I}(\mathcal{A}')}$. Since the solution is correct, we know that $x^{\mathcal{I}(\mathcal{A}')} \in succ(c)^{\mathcal{I}(\mathcal{A}')}$.

Since we know that $\mathcal{I}(\mathcal{A}')$ satisfies $\mathcal{A}'$ and that $\mathcal{A} \subseteq \mathcal{A}'$, $\mathcal{I}(\mathcal{A}')$ also satisfies $\mathcal{A}$. □

Lemma 4. If $\mathcal{A} := \{x : C\}$ is satisfiable then the Tableau-algorithm terminates without a clash.

Proof. By Lemma 1 we know that the algorithm always terminates. It remains to show that the algorithm terminates with a clash-free ABox. Since $\mathcal{A}$ is satisfiable it does not contain a clash. Let $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ be an interpretation which satisfies $\mathcal{A}$. We show that if $\mathcal{A}_i$ does not contain a clash and $\mathcal{I}$ satisfies $\mathcal{A}_i$, then $\mathcal{A}_{i+1}$ can be obtained from $\mathcal{A}_i$ by applying a rule while maintaining clash-freeness and satisfiability by $\mathcal{I}$.
We already stated that $\mathcal{I}$ satisfies the clash-free $\mathcal{A} =: \mathcal{A}_0$ (induction base). Let $\mathcal{A}_i$ be a clash-free ABox which is satisfied by $\mathcal{I}$ (induction hypothesis). We distinguish the cases based on the rules we apply on $\mathcal{A}_i$ to obtain $\mathcal{A}_{i+1}$ (induction step):

- we apply the $\sqcap$-rule on $x : C \sqcap D$: The interpretation $\mathcal{I}$ satisfies $\mathcal{A}_{i+1} = \mathcal{A}_i \cup \{x : C, x : D\}$ because by the hypothesis $\mathcal{I}$ already satisfies $\mathcal{A}_i$ and hence also $x : C \sqcap D$. That means that $x^{\mathcal{I}} \in (C \sqcap D)^{\mathcal{I}}$ and therefore $\{x, C, x : D\} \cup \mathcal{A}_i$ is satisfied by $\mathcal{I}$

- we apply the $\sqcup$-rule on $x : C \sqcup D$: We have to show that either $\mathcal{A}_{i+1} = \mathcal{A}_i \cup \{x : C\}$ or $\mathcal{A}_{i+1} = \mathcal{A}_i \cup \{x : D\}$ is satisfied by $\mathcal{I}$. Again by the induction hypothesis $\mathcal{A}_i$ is satisfied by $\mathcal{I}$ and hence $x^{\mathcal{I}} = (C \sqcup D)^{\mathcal{I}}$. So either $x^{\mathcal{I}} \in C^{\mathcal{I}}$ and hence we choose $\mathcal{A}_{i+1} = \mathcal{A}_i \cup \{x : C\}$ or $x^{\mathcal{I}} \in D^{\mathcal{I}}$ and hence we choose $\mathcal{A}_{i+1} = \mathcal{A}_i \cup \{x : D\}$. In both cases $\mathcal{I}$ satisfies $\mathcal{A}_{i+1}$.

- we apply the *succ*-rule on $x : succ(c)$: We have to show that by this step we are able to add successors such that $\mathcal{I}$ satisfies $\mathcal{A}_{i+1}$. In this step, we gather first all *succ*-assertions together in a set $\mathcal{S}$, formulate a QFBAPA formula $\phi(x)$ and let a solver return us all possible solutions within an upper bound. Because $\mathcal{A}_i$ is satisfiable, $\mathcal{S}$ is also satisfiable (subset of $\mathcal{A}_i$). Hence there have to be solutions which can be returned by the solver. We need to show that the solver is capable of returning a solution within our upper bound, such that $\mathcal{A}_{i+1}$ is satisfied by $\mathcal{I}$. In case $x^{\mathcal{I}}$ has no successors, the empty solution must be a valid solution, which can be returned from the solver. If $\mathcal{I}$ is finite and the number of x's successors within each Venn region is within our upper bound, then we can create a solution σ induced by $\mathcal{I}$, which can be returned by our solver. In any other case we have to show that we can create a (finite) solution from $\mathcal{I}$, which the solver is able to return. We know that $x^{\mathcal{I}}$ must have a finite number of successors in $\mathcal{I}$. Therefore we can create a solution σ based on that: Let σ be an empty solution. For each $y^{\mathcal{I}} \in \bigcup_{r \in \mathbf{R}} r^{\mathcal{I}}(x)$ we introduce an element e and add it to $\sigma(\mathcal{U})$:

  - for each $(x^{\mathcal{I}}, y^{\mathcal{I}}) \in r^{\mathcal{I}}$ add e to $\sigma(X_r)$
  - for each $y^{\mathcal{I}} \in C^{\mathcal{I}}$ add e to $\sigma(X_C)$
  - for each $y^{\mathcal{I}} \in succ(c)^{\mathcal{I}}$ add e to $\sigma(X_c)$

It is clear that if the solver returns this solution, then $\mathcal{I}$ satisfies $\mathcal{A}_{i+1}$. If each Venn region of this solution has more elements than the calculated upper bound, we can reduce the number of successors by the following steps [5]: Convert the QFBAPA formula to a system of linear equations $An = b$ like in Section 3.2. Let n be a solution to this system such that $n_k = |\{e|e \in \sigma(X_1^i) \cap \cdots \cap \sigma(X_m^i)\}|$, where $X_1^i \cap \cdots \cap X_m^i$ is the k-th Venn region. Then we can reduce n to n' like shown in Section 3.2. With the help of n' we create the new solution σ' by adding n'_k successors in the k-th Venn region. It holds that (†):

- $\sigma^*(\mathcal{U}) \subseteq \sigma(\mathcal{U})$
- for each $e \in \sigma^*(\mathcal{U})$: $e \in \sigma^*(X_v)$ iff $e \in \sigma(X_m)$ with $m \in \mathbf{C} \cup \mathbf{R} \cup \{succ(c)|x : succ(c) \in \mathcal{A}_i\}$

The reason is the fact that we decrease the number of successors in specific Venn-regions to obtain n'.
The algorithm then creates individual names according to the solution σ', which leads to the satisfaction of all *succ*-assertions of x. For each element $e \in \sigma'(\mathcal{U})$ we know that there is an individual name y_e such that:

- $y_e : C \in \mathcal{A}_{i+1}$ iff $e \in \sigma'(X_C)$, $C \in \mathbf{C}$
- $y_e : succ(c) \in \mathcal{A}_{i+1}$ iff $e \in \sigma'(X_{succ(c)})$
- $(x, y_e) : r \in \mathcal{A}_{i+1}$ iff $e \in \sigma'(X_r)$

Because of the fact in (†) we can conclude

- $y_e : C \in \mathcal{A}_{i+1}$ iff $e \in \sigma(X_C)$, $C \in \mathbf{C}$
- $y_e : succ(c) \in \mathcal{A}_{i+1}$ iff $e \in \sigma(X_{succ(c)})$
- $(x, y_e) : r \in \mathcal{A}_{i+1}$ iff $e \in \sigma(X_r)$

Since σ is induced by $\mathcal{I}$:

- $y_e : C \in \mathcal{A}_{i+1}$ iff $y^{\mathcal{I}} \in C^{\mathcal{I}}$, $C \in \mathbf{C}$
- $y_e : succ(c) \in \mathcal{A}_{i+1}$ iff $y^{\mathcal{I}} \in succ(c)^{\mathcal{I}}$
- $(x, y_e) : r \in \mathcal{A}_{i+1}$ iff $(x^{\mathcal{I}}, y^{\mathcal{I}}) \in r^{\mathcal{I}}$

Hence we can extend $\mathcal{I}$ by $y_e := y^{\mathcal{I}}$ which satisfies $\mathcal{A}_{i+1}$

□

Chapter 5

Conclusion

blub

Bibliography

[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] V. Kuncak and M. Rinard. Towards efficient satisfiability checking for Boolean Algebra with Presburger Arithmetic. In *Conference on Automateded Deduction (CADE-21)*, volume 4603 of *LNCS*, 2007.

[5] C. H. Papadimitriou. On the complexity of integer programming. *J. ACM*, 28(4):765–768, Oct. 1981.

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

[7] 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.