TECHNISCHE UNIVERSITÄT DRESDEN

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

December 2, 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 signature

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

1.1 Motivation

In traditional databases stored data objects do not have any relation with each other unless explicitly stated. However we can extract additional information about these objects if we use database systems which employ *semantics*. Imagine we want to add data entries for two people (Anna and Beth) to a traditional database. Anna is a teacher at the local school. Beth is a student of her class. In our database we save their names, the class that Beth attends as well as the class that Anna teaches. As we do not explicitly encode their student-teacher relationship the traditional database does not know about it. If we use an ontology-based system we can deduce this information by making use of semantics. These semantics are described by a set of rules (axioms). In our example one axiom would be that if a teacher teaches a class and students attend the same class, then there is a student-teacher relationship between the teacher and these students. By applying this axiom the ontology-based database can automatically deduce that Anna is the teacher of Beth. Popular use-cases for ontology-based systems are databases for biological and medical research [4]. As an example ontologies can be used to automatically fill in missing information about patients which are helpful in diagnostics. Another major use-case for ontologies is the *Semantic Web*, which is an extension of the World Wide Web with standards given by the World Wild Web Consortium (W3C)[1]. These standards allow a more effective way of combining information from different sources.
An Ontology (in the field of computer science) can be viewed as formal representation of a certain domain of interest. The relationships between entities in an ontology-based database are formulated by a fragment of first-order logic (FOL). This fragment of FOL is called *Description Logic (DL)* and is a family of knowledge representation systems. DLs mainly consist of concepts, which correspond to unary relations in FOL, and relations between the concepts, which

[1]`https://www.w3.org/standards/semanticweb/`, last accessed on December 2, 2020

correspond to binary relations in FOL. To create more complex (compound) concepts we can combine concepts by using operators like $\sqcap$, $\sqcup$, $\sqsubseteq$, $\exists$ and $\forall$. For example the statement "All Humans who have children are parents" can be formalized in DL as $Human \sqcap \exists hasChild.\top \sqsubseteq Parent$, where $Human$ and $Parent$ are concept names and $hasChild$ is a role name. This statement can also be formalized with a a numerical restriction: $Human \sqcap \geq 1 hasChild.\top \sqsubseteq Parent$. A knowledge base consists of a *TBox*, which contains the axioms, and an *ABox*, which contains assertions about certain individual names (objects).
The process of determining whether some statement can be concluded from a set of information is called *reasoning*. Reasoning can be done by adding this statement in negated form (as an axiom or assertion) to the set of information (TBox or ABox) and then checking whether the updated knowledge base is now *unsatisfiable*. If it is unsatisfiable the statement can be concluded from the information set. Being able to check the satisfiability of DL statements is therefore a valuable tool to conduct reasoning in ontology systems. The DL $\mathcal{ALCQ}$ [5][9] has been investigated thoroughly and therefore we know a lot about its satisfiability. This DL allows conjunctions ($\sqcap$), disjunctions ($\sqcup$), negations ($\neg$) and number restrictions ($\leq n\,r\,C$ and $\geq n\,r\,C$, where n is a number, r a role name and C a concept name). The authors of [5] proved that checking satisfiability of a $\mathcal{ALCQ}$ concept without a TBox is in PSpace and otherwise in ExpTime.
The DL $\mathcal{ALCSCC}$ [1] extends $\mathcal{ALCQ}$ with *set constraints* and *cardinality constraints* over role successors, which use the logic of *QFBAPA* (quantifier-free fragment of Boolean Algebra with Presburger Arithmetic)[7]. Instead of the quantifiers $\exists$ and $\forall$ 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 quantified restriction of the form $\leq n\,r.C$ or $\geq n\,r.C$ can be written in $\mathcal{ALCSCC}$ as $succ(|r \cap C| \leq 1)$ or $succ(|r \cap C| \geq 1)$ respectively. However a constraint like $succ(|r| = |s|)$ can not be formulated in $\mathcal{ALCQ}$ [1]. Because of this extension checking satisfiability over $\mathcal{ALCSCC}$ becomes more complicated. Nevertheless the author in [1] shows that the satisfiability problem for $\mathcal{ALCSCC}$ is still PSpace-complete.
In this work we present a tableau algorithm for $\mathcal{ALCSCC}$. While a tableau algorithm leads to a runtime complexity worse than PSpace-complete the benefit of a tableau algorithm is that we gain a satisfied interpretation (a correct assignment without contradiction) of the concepts, which is also called *witness*. A tableau algorithm consists of completion rules which are applied to the assertions of the ABox. By applying these rules new assertions that can be derived from the original assertions are added to the ABox. If we can no longer apply any rules and the ABox contains a contradiction, then the ABox is unsatisfiable. Otherwise it is satisfiable. The main difficulty in creating the completion rules for $\mathcal{ALCSCC}$ is that unlike in $\mathcal{ALCQ}$, the number of successor is not bounded. By adding role successors the cardinalities in a constraint can vary. For example if we have a constraint $succ(|r| = |s|)$ then the bound for the number of s-successors is equal to the number of r-successors we already have. During a tableau algorithm we can add, merge or replace r-successors which changes the

bound for the number of s-successors. To deal with the changing cardinalities of $\mathcal{ALCSCC}$ we transform the assertions with cardinalities to a QFBAPA formula and use a QFBAPA solver to determine whether the formula is satisfiable or not. If the formula is not satisfiable, there must be a contradiction. If the solver returns a solution, we add new assertions accordingly. Since we use a solver that is capable of returning every possible solution, there can be an infinite number of solutions. We can show that we can shorten each of these solutions to a bound number of role successors without losing any information.

1.2 Related Works

In [5] a *tableau* algorithm is presented for checking satisfiability for an ABox in the DL $\mathcal{ALCQ}$. The satisfiability (of concepts) is stated in [5] as a PSpace-hard problem. The algorithm in [5] consists five rules which can be applied non-deterministically and in any order. Two of the rules are decomposing rules for $\sqcap$ and $\sqcup$. Another rule determines whether a successor of an individual name also contributes in any other numerical assertion, which helps to determine the exact number of each role successor. The last two rules are able to add and replace individual names. The twisted part here is, that an unless loop of adding and replacing invidual names is possible. therefore the author introduce a concept of *safeness*, which has a similar purpose as blocking in other tableau algorithm. In [10] 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 [5], 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*.
In [6] two tableau algorithm are presented for $\mathcal{SHIF}$ concepts and $\mathcal{SI}$ concepts. The DL $\mathcal{SI}$ extends the DL $\mathcal{ALC}$ with transitive and inverse role names. The DL $\mathcal{SHIF}$ extends $\mathcal{SHIF}$ with role hierarchy and functional restriction. For both DL the tableau algorithms have to have some blocking technique to avoid infinite chain of introducing elements with the same assertions which can result from the transitive roles and inverse roles. For $\mathcal{SI}$ the tableau algorithm consider not only the successor but also the predecessor of the considered individual name whenever considering $\forall$-assertions. In case of $\exists$-assertions the algorithm first determinates whether the considered individual name x is blocked or not. It is blocked when an ancestor (not directly a predecessor) is blocked or an ancestor has "similar" assertion as x. This algorithm runs in PSpace. For $\mathcal{SHIF}$ the tableau algorithm have to consider for every rule that the considered individual name x is not pair-wise blocked. Being pair-wise blocked means that for an predecessor y of x there are two ancestora of x, such that they behave "similar" to y and x. in [3] a DL called $\mathcal{SHQ}$ (also known $\mathcal{ALCQH}_{R^+}$ from the SI family is presented. This DL does not have inverse role, but role hierarchy and numerical restriction. The problem here is that with numerical restriction an infinite chain of adding individual names can avoid the termination of the tableau algorithm. Hence a blocking technique is also used: An individual name

x blocks another individual name y if x was introduced before y and if any assertion about x holds for y, too. To deal with the number restriction a reasoner about set of linear inequations is used.
Regarding the DL $\mathcal{ALCSCC}$, which is considered in this work, [1] gives a solution for the satisfiability problem without TBox, which has the complexity PSpace-complete: 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. For the satisfiability problem with a TBox a type elimination algorithm is presented.

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 [7] 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)

In [7] it is shown that checking satisfiability of QFBAPA formulas is a NP-complete problem.

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.

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

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 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 [5],[9]. The first four rules on the right hand side and the rules $C^{\neg} \rightarrow \neg C$ 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 similarly to $\neg(C \sqcap D) \rightarrow \neg C \sqcup \neg D$ and $\neg(C \sqcap D) \rightarrow \neg C \sqcup \neg D$. 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 $y \in (C \cap D)^{\mathcal{I}_x}$:

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

Next we define the size of an $\mathcal{ALCSCC}$ concept C inductively over concepts, set terms and cardinality constraints. The size is important for the termination proof.

- $size(r) = size(C) = 1$ if $r \in \mathbf{R}$, $C \in \mathbf{C}$
- $size(n) = size(|k|) = 1$ if $n \in \mathbb{N}$, k cardinality term
- $size(\neg C) = size(C) + 1$
- $size(k^{\neg}) = size(k) + 1$
- $size(C \sqcap D) = size(C \sqcup D) = size(C) + size(D) + 1$
- $size(k \cap l) = size(k \cup l) = size(k) + size(l) + 1$
- $size(|k|) = size(k)$
- $size(succ(c)) = \begin{cases} 1 + size(k) + size(l) & c = k \leq l \\ 1 + size(l) & c = n\, dvd\, l \end{cases}$

Chapter 3

Tableau for $\mathcal{ALCSCC}$

To check whether a concept is satisfiable one can use a tableau algorithm. Even though the complexity of a tableau algorithm can grow exponentially the advantages of it is that it offers a possible satisfied interpretation, a *witness*, for the given ABox if satisfiable. 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. Some rules gives the algorithm a choice to make like in case of disjunctions. 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 $\mathcal{A} = \{x : C\}$. We use the induced interpretation again for the soundness proof.
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 $x^{\mathcal{I}(\mathcal{A})} \notin succ(c)^{\mathcal{I}(\mathcal{A})}$.

Violated assertions are not necessary a clash, but if we can not apply any rules any more then we do not have the possibility to lift the violation. Therefore we define such situation *also* as a clash, beside obvious contradictions.

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

One major difficulty of $\mathcal{ALCSCC}$ are the numerical arithmetic, the successor-assertion. Not only because the number of successors in a constraint can vary, but also in combination with *nested levels* which denotes in which "generation" a successor lays. Recall that we describe the tableau algorithm to decide the satisfiability for an concept.

Definition 9 (Nested Level)**.** Let $\mathcal{A} = \{x : C\}$ be an ABox. An individual name lays in the i-th nested level if it is the i-th individual name in a role chain beginning from x. The individual name x is in the 0th nested level. Every successor of x are in the 1st nested level.

In some DLs we are able to describe the successor of a successor like $\exists r.(\exists r.C)$. But again in $\mathcal{ALCSCC}$ the bond of number for such assertion can vary. 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 next nested level, which means we consider only the direct successors of x first. We assume that the ABox is already in NNF. As an example for the transformation we look at

Example 1.

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

with $\mathbf{C} = \{A, B, C\}$ and $\mathbf{R} = \{r\}$

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 individual names for $succ(|A| \leq |B \cap r|)$ are on a different nested level than the ones for $succ(|A| \leq |B|)$. 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 first gather all *succ*-assertion regarding x together and transform the cardinality constraint 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 can now let the solver gives us a possible solution, if there is one. For the QFBAPA solver we take two assumption about it such that we can work with it in the tableau algorithm.

Assumption 1. We assume that every considered QFABAPA solver is correct which means

- it terminates for all QFBAPA formulas (which are finite)
- the formula is satisfiable iff it returns a solution

Assumption 2. Let ϕ be an arbitrary QFBAPA formula. We assume that every considered QFABAPA solver is able to return all possible solutions of ϕ.

3.2 Solution of a formula

Since we make Assumption 2 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 can work on an infinite space and/or might not terminate. Therefore we want to only consider solutions inside some pre-computed *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 [8], which solution we are going to use. 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

$$\phi' = |X_c| \geq 1 \wedge |X_A| - |X_B| \leq 0 \wedge |\mathcal{U} \cap X_r^{\neg}| = 0 \wedge |\mathcal{U}^{\neg} \cap X_r| = 0 \tag{3.2}$$

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

$$\begin{aligned}\phi'' =& |X_c| - I_1 = 1 \wedge |X_A| - |X_B| + I_2 = 0 \wedge \\ & |\mathcal{U} \cap X_r^\neg| = 0 \wedge |\mathcal{U}^\neg \cap X_r| = 0 \end{aligned} \tag{3.3}$$

Right now it is not clear whether the set variables are overlapping or not which is a problem because for a system of linear equations the variables are always disjunct. 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. Since we have 5 set variable, we have $2^5 = 32$ Venn regions. 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. Therefore we construct A and b such that instead of assigning elements to set variables we assign them to the remaining 8 Venn regions. That means for the vector x the entry x_k, $1 \leq k \leq 8$, denotes the number of elements in the k-th Venn region. Since we have four equations and two slack variables the matrix A has four rows and 10 columns with a_{ij}, $1 \leq i \leq 4$ and $1 \leq j \leq 10$, denoting the sum of the pre-factors of the set variables, in which the j-th Venn region is included, occurring in the i-th equation. 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 i-th Venn region is empty. The last two rows of A, which present the equations $|\mathcal{U} \cap X_r^\neg| = 0$ and $|\mathcal{U}^\neg \cap X_r| = 0$, 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 [8] can be used to establish an upper bound for the solution of the *ILP* in NP:

Theorem 1 (Theorem 1 from [8])**.** 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, which is important for the termination proof.

In our example the upper bound for all x_i is $10 \cdot (4 \cdot max\{|1|, |-1|\})^{2 \cdot 4+1} = 2621440$, which means that we can discard any solution in which a Venn region has more than 2621440 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. That means we want to consider conjunctions and disjunctions first and then consider the transformation to a QFBAPA formula. 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. In the second part 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 10 (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 $x : succ(c)$, this rule has not been applied for x yet and no boolean rules are 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 $\mathcal{A} := \mathcal{A} \cup \{x :\perp\}$
  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 set $\mathcal{A} = \mathcal{A} \cup \{y_e : succ(c)\}$
  - 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 2621440 (see Section 3.2). If the formula have been unsatisfiable, we would add $x :\perp$ to our ABox. this would have led us to the end of the tableau algorithm because this "choice" is the only one possible since we do not have any disjunction. However since the formula is satisfiable we obtain a solution from the solver because of Assumption 1. 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. The considered QFBAPA formula is capable of returning this solution because of Assumption 2. Obviously in this solution every Venn-region has less element than 2621440. 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 $\mathcal{A}$.

$$\begin{tikzcd}
 & & C & & \\
 & & x \arrow[dl, "r"'] \arrow[dr, "r"] & & \\
B & y_e & & y_f & succ(|A| \leq |B \cap r|)
\end{tikzcd}$$

Then for y_f we have to apply the *successor*-rule because no boolean rule is applicable. Since $y_f : succ(|A| \leq |B \cap r|)$ is the only *succ* assertion we only have to add an r-successor of the concept B. 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 $\mathcal{A}$.

$$\begin{tikzcd}
 & & C & & \\
 & & x \arrow[dl, "r"'] \arrow[dr, "r"] & & \\
B & y_e & & y_f \arrow[d, "r"] & succ(|A| \leq |B \cap r|) \\
 & & & y_g & B
\end{tikzcd}$$

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 the following:

- 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 consider Assumption 1 and 2. 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 $\mathcal{A}_1$. Each rule terminates:

- $\sqcap$- and $\sqcup$-rule: Obviously both rule terminates because we decompose finite compound concepts.
- *successor*-rule: Since ABoxes are finite we can only have a finite subset $\mathcal{S}$ of *successor*-assertions and therefore can always form a (finite) QFBAPA formula. By Assumption 1 the QFBAPA solver always terminates and we gain a finite solution. Therefore we always add a finite number of successor in this rule. Hence this rule application always terminates.

For the termination proof we map every ABox $\mathcal{A}$ to an element $\Psi(\mathcal{A})$ of a set Q. Each $\Psi(\mathcal{A})$ consists of triples $\psi_{\mathcal{A}}(x)$ for each individual name x. Let $\prec$ be an strict partial ordering, which is a irreflexive (if $a \prec b$ then $b \not\prec a$) and transitive (if $a \prec b$ and $b \prec c$ then $a \prec c$) relation (with a, b, c being comparable elements). If we show that $\prec$ is well-founded, e.g. there is no infinite decreasing chain, and that for every ABox $\mathcal{A}'$, which is derivable from an ABox $\mathcal{A}$, we can only have $\Psi(\mathcal{A}') \prec \Psi(\mathcal{A})$ then the termination of the algorithm can be concluded.

Each triple in Q consists of a multiset of natural numbers and two natural numbers. A multiset of natural numbers M is smaller than another multiset of natural numbers M' if we can obtain M from M' by replacing at least one number n of M' with a set of natural numbers, which are all smaller than n, or by deleting at least the number n. For example $M = \{2,2,2,1,5\}$ is smaller than $M' = \{2,3,5\}$ because the second entry $\{3\}$ of M' can be replaced by $\{2,2,1\}$. We say that the empty mutliset $\{\}$ is always smaller than any multiset of natural numbers. Since the multisets consist of natural numbers, which can be ordered by the strict partial order $<$, they can be ordered by $\prec$, too. A triple $T_1 = (x_1, x_2, x_3)$ is smaller than a triple $T_2 = (y_1, y_2, y_3)$ if T_1 is lexicographically smaller (from right to left) than T_2 which means that for the first $i \in \{1,2,3\}$ such that $x_i \neq y_i$ it holds that x_i is smaller than y_i. Again because we have triples of numbers they can be ordered by $\prec$, too. Therefore $\Psi(\mathcal{A}') \prec \Psi(\mathcal{A})$ if we can replace one triple $\psi_{\mathcal{A}}(x)$ in $\Psi(\mathcal{A})$ with at least one triple $\psi_{\mathcal{A}'}(x')$, such that $\psi_{\mathcal{A}}(x) \prec \psi_{\mathcal{A}'}(x')$ to obtain $\Psi(\mathcal{A}')$ or if we remove at least one triple $\psi_{\mathcal{A}}(x)$. Note that we assume earlier in Section 2.2 that each $C \sqcap D$ and $C \sqcup D$ in a cardinality term is replaced by $C \cap D$ and $C \cup D$ respectively.
We describe now how the triples in Q looks like.

Definition 11. 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 $\psi_{\mathcal{A},1}(x)$ is the natural number $max\{size(C) | x : C \in \mathcal{A}\}$.
- The second component $\psi_{\mathcal{A},2}(x)$ is a multiset which contains for each assertion $x : C \sqcap D \in \mathcal{A}$ for which the $\sqcap$-rule is applicable the natural number $size(C \sqcap D)$. Respectively for $x : C \sqcup D$.
- The third component $\psi_{\mathcal{A},3}(x)$ is the number 1 if the *successor*-rule is applicable, 0 otherwise.

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

$$\Psi(\mathcal{A}) = \{\psi_{\mathcal{A}}(x)\} = \{(7, \{7,7\}, 0)\} \tag{4.1}$$

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

$$\Psi(\mathcal{A}^1) = \{\psi_{\mathcal{A}^1}(x)\} = \{(7, \{\}, 2)\} \tag{4.2}$$

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

$$\Psi(\mathcal{A}^2) = \{\psi_{\mathcal{A}^2}(x), \psi_{\mathcal{A}^2}(y_e), \psi_{\mathcal{A}^2}(y_f)\} = \{(7, \{\}, 0), (1, \{\}, 0), (3, \{\}, 1)\} \tag{4.3}$$

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

$$\begin{aligned}\Psi(\mathcal{A}^3) = \{\psi_{\mathcal{A}^3}(x), \psi_{\mathcal{A}^3}(y_e), \psi_{\mathcal{A}^3}(y_f), \psi_{\mathcal{A}^3}(y_g)\} = \\ \{(7, \{\}, 0), (1, \{\}, 0), (3, \{\}, 0), (2, \{\}, 0)\}\end{aligned} \tag{4.4}$$

The newly introduced triple $\psi_{\mathcal{A}^3}(y_g)$ is smaller then $\psi_{\mathcal{A}^2}(y_f)$, because $\psi_{\mathcal{A}^3,1}(y_g) < \psi_{\mathcal{A}^2,1}(y_f)$, and hence we have $\Psi(\mathcal{A}^3) \prec \Psi(\mathcal{A}^2) \prec \Psi(\mathcal{A}^1) \prec \Psi(\mathcal{A})$.
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 derivable 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 on $x : C \sqcap D$: The first component remains unchanged because $size(C) \leq size(C \sqcap D)$ and $size(D) \leq size(C \sqcap D)$. We have $\psi_{\mathcal{A}',2}(x) \prec \psi_{\mathcal{A},2}(x)$ 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 $\psi_{\mathcal{A}',2}(x)$ still becomes smaller because $size(C) < size(C) + size(D) + 1 = size(C \sqcap D)$ (respectively for $size(D)$). Hence $\psi_{\mathcal{A}'}(x) \prec \psi_{\mathcal{A}}(x)$.
  For any other individual name y, such that $y \neq x$, the triple $\psi_{\mathcal{A}}(y)$ remains unchanged.

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

- $\mathcal{A}'$ is obtained from $\mathcal{A}$ by applying the *successor*-rule on $x : succ(c)$: $\psi_{\mathcal{A},1}(x)$ remains unchanged because we do not add any assertion for x. We are able to apply this rule, because both $\sqcap$-rule and $\sqcup$-rule are not applicable on $\mathcal{A}$ and we do not have applied this rule for x yet. Because we do not add assertions for x and do not decompose any disjunction or conjunction we know that $\psi_{\mathcal{A},2}(x)$ remains unchanged. We also know that $\psi_{\mathcal{A},3}(x) = 1$ because we are able to apply the *successor*-rule on an assertion of x. Afterwards we have $\psi_{\mathcal{A}',3}(x) = 0$. Therefore $\psi_{\mathcal{A}'}(x) \prec \psi_{\mathcal{A}}(x)$.
  For every freshly introduced individual name y we have to add a triple $\psi_{\mathcal{A}'}(y)$ to $\Psi(\mathcal{A}')$. 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 $size(succ(c)) > size(C)$. That means 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 $size(C)$. Therefore $\psi_{\mathcal{A}',1}(y) < \psi_{\mathcal{A}',1}(x)$ and hence $\psi_{\mathcal{A}'}(y) \prec \psi_{\mathcal{A}'}(x)$.
  For any other individual name z, such that $z \neq x$ and $z = y$, 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 new triples are always smaller than $\psi_{\mathcal{A}'}(x)$. Therefore $\Psi(\mathcal{A}') \prec \Psi(\mathcal{A})$. Because we work with natural numbers the ordering $<$ over them is well-founded. Therefore we also know that $\prec$ multisets over natural numbers is also well-founded [2](Theorem 2.5.5). Since the natural numbers and multisets of natural numbers can be ordered by a well-founded ordering, the triples after Definition 11, which are "lexicographical products of two terminating relation" [2](Theorem 2.4.2), can be ordered by a well-founded ordering as well. Therefore $\prec$ over the multisets of these triples is well-founded, too [2](Theorem 2.5.5). $\square$

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.
We start with one direction of the *iff* statement.

Lemma 3 (Soundness)**.** *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. From Lemma 2 we know that the tableau algorithm always terminates. 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 $\mathcal{A}$. □

We now proof that we can return any ABox such that we can induce every possible model within the pre-computed upper bound.

Lemma 4 (Completeness)**.** 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 $e \in \bigcup_{r \in \mathbf{R}} r^{\mathcal{I}}(x)$ we add e to $\sigma(\mathcal{U})$:

- for each $(x^{\mathcal{I}}, e) \in r^{\mathcal{I}}$ add e to $\sigma(X_r)$
- for each $e \in C^{\mathcal{I}}$ add e to $\sigma(X_C)$
- for each $e \in succ(c)^{\mathcal{I}}$ 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 [8]: Convert the QFBAPA formula to a system of linear equations $An = b$ like in Section 3.2. We know there has to be a solution, because $\mathcal{S}$ is satisfiable. 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 $e \in C^{\mathcal{I}}$, $C \in \mathbf{C}$
- $y_e : succ(c) \in \mathcal{A}_{i+1}$ iff $e \in succ(c)^{\mathcal{I}}$
- $(x, y_e) : r \in \mathcal{A}_{i+1}$ iff $(x^{\mathcal{I}}, e) \in r^{\mathcal{I}}$

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

$\square$

Chapter 5

Conclusion

The description logic $\mathcal{ALCSCC}$ extends the well-known description logic $\mathcal{ALCQ}$ with set constraints and cardinality constraints over role successors which are hard to deal with when checking satisfiability over $\mathcal{ALCSCC}$. We present a tableau algorithm which uses a QFBAPA solver to deal with the successors. It is shown that checking satisfiability for this DL is in PSpace, however the tableau algorithm runs in 2ExpSpace: We know that in each *successor*-rule application we can add in worse case exponential many successors as shown in Section 3.2. Each of the newly added successors is also capable of obtaining exponential many successors. The advantage of this approach is that we do not only return the statement about the satisfiability but also a satisfied interpretation, a model, for the concept.
For future works one can extend the algorithm for $\mathcal{ALCSCC}$ concepts w.r.t. a TBox. In [1] it is stated that the satisfiability problem w.r.t. a TBox is in ExpTime. One approach can be to add all information we can concluded from the TBox first i.e. if we have $C \sqsubseteq D$ (says that every individual name in C is also in D) and $x : C$ then $x : D$ has to be added to the ABox, too. We have do the same for every freshly introduce individual name, too. One can also research for a tableau algorithm which does not use a QFBAPA solver. For that introducing blocking technique is most likely required because like in for $\mathcal{ALCQ}$ and the $\mathcal{SI}$ families endless loops of adding and merging (or replacing) individual names are possible. Another interesting area to do research on is the pre-computed upper bound. In this work the bound is exponentially large which is the reason for the ExpSpace complexity. If there exists a smaller upper bound one can run this algorithm in a smaller complexity.

Bibliography

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

[2] F. Baader and T. Nipkow. *Abstract Reduction Systems*, page 7–33. Cambridge University Press, 1998.

[3] V. Haarslev. Combining tableau and algebraic methods for reasoning with qualified number restrictions in description logics. 05 2002.

[4] 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.

[5] B. Hollunder and F. Baader. Qualifying Number Restrictions Concept Languages. Technical report, Research Report RR-91-03, 1991.

[6] I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for very expressive description logics. *CoRR*, cs.LO/0005014, 05 2000.

[7] 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.

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

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

[10] 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.