European Master's Program in
Computational Logic (EMCL)

Fakultät Informatik
Technische Universität Dresden

Integrating Reasoning Services for Description Logics with Cardinality Constraints with Numerical Optimization Techniques

Master's Thesis

April 2019

Author
Filippo De Bortoli
MNr.: 4732669

Supervisor
Prof. Dr.-Ing. Franz Baader

Co-supervisor
Prof. Rafael Peñaloza Nyssen

A document submitted in partial fulfillment of the requirements for the degree of
Master of Science in *Computational Logic*

Declaration of Authorship

Author: Filippo De Bortoli
Immatriculation number: 4732669
Title: Integrating Reasoning Services for Description Logics with Cardinality Constraints with Numerical Optimization Techniques

I hereby declare that except where specific reference is made to the work of others, the contents of this thesis are original and have not been submitted in whole or in part to obtain a degree or any other qualification neither in this nor in any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text. No other resources apart from the references and auxiliary means indicated in the bibliography were used in the development of the presented work.

_____ _____

(Place and date) (Author's signature)

Abstract

Recent research in the field of Description Logic (DL) investigated the complexity of the satisfiability problem for description logics that are obtained by enriching the well-known DL $\mathcal{ALCQ}$ with more complex set and cardinality constraints over role successors. The algorithms that have been proposed so far, despite providing worst-case optimal decision procedures for the concept satisfiability problem (both without and with a terminology) lack the efficiency needed to obtain usable implementations. In particular, the algorithm for the case without terminology is nondeterministic and the one for the case with a terminology is also best-case exponential. The goal of this thesis is to use well-established techniques from the field of numerical optimization, such as column generation, in order to obtain more practical algorithms. As a starting point, efficient approaches for dealing with counting quantifiers over unary predicates based on SAT-based column generation should be considered.

Contents

1 Introduction

Description Logics (DLs) [4] are a class of logic-based formalisms for knowledge representation. They are used as specification languages for ontologies in several areas of application, for example in medicine [18]. The information defined in a DL about a particular domain of discourse is built upon predicates that describe classes of features with *concept names* and relationships between elements in the ontology using *role names*. A description of an element is then obtained as a Boolean combination of these predicates, resulting in a formal concept.

For example, the concept of a pizza that has at least a topping that is a vegetable and has no pineapple on it can be formalized by the concept description

$$\texttt{Pizza} \sqcap \exists \texttt{topping.Vegetable} \sqcap \forall \texttt{ingredient.} \neg \texttt{Pineapple},$$

which uses the concept names `Pizza`, `Vegetable` and `Pineapple` and the role names `topping` and `ingredient`, together with the concept constructors conjunction ($\sqcap$), negation ($\neg$), existential restriction ($\exists r.C$) and value restriction ($\forall r.C$).

Qualified number restrictions, introduced in the well-studied DL $\mathcal{ALCQ}$ [4], enable the specification of quantitative characteristics in a DL. In their simplest form, these restrictions convey information about the number of role successors of an individual, together with the description that captures those successors. For instance, the concept of a light and tasty pizza, intended as a pizza with at least three ingredients but at most two meat-based ingredients, can be formalized as

$$\texttt{Pizza} \sqcap (\geq 3 \ \texttt{ingredient.} \ \top) \sqcap (\leq 2 \ \texttt{ingredient.} \ \texttt{Meat})$$

The class of constraints over role successors that are expressible in the logic QFBAPA [15], encompassing the ones expressible in $\mathcal{ALCQ}$, has been first introduced in the field of Description Logics with the DL $\mathcal{ALCSCC}$ [2]. In this setting, the constraints are generalized and defined using arbitrary Boolean combinations of role and concept names; in $\mathcal{ALCQ}$, the number restrictions have a clear syntactic form that, for instance, prevents one from encoding information about the relationships between different role names. An example of a concept that can be expressed using the constructors of $\mathcal{ALCSCC}$ is

$$\texttt{Pizza} \sqcap \mathrm{succ}(\texttt{preparedBy} \subseteq \neg \texttt{eatenBy}) \sqcap \mathrm{succ}(|\texttt{topping} \cap \texttt{Vegetable}| = |\texttt{topping} \cap \texttt{Meat}|)$$

which states that every pizza described by this concept is not eaten by any person that prepared it and that it has the same number of meat-based and plant-based toppings.

The complexity of reasoning in the DL $\mathcal{ALCSCC}$ has been studied in [2]. In particular, the problem of *concept satisfiability* — checking whether there are individuals that can be categorized using a given concept description — has been thoroughly analyzed. The case where concept satis-

fiability is tested without a supporting terminological knowledge base (also called TBox) has been classified as a PSPACE-complete problem. This is a positive result, because it shows that although the DL $\mathcal{ALCSCC}$ is more expressive than $\mathcal{ALCQ}$, the reasoning complexity remains unchanged — $\mathcal{ALCQ}$ concept satisfiability without a TBox is also PSPACE-complete [4]. However, the algorithm devised in [2] is not suitable for practical purposes. One reason is that its specification employs several forms of non-deterministic guessing during its execution.

The idea of applying efficient and proven methods to deploy quantitative reasoning in DLs with quantified number restrictions is not a novel one and has already been investigated using different methodologies. Promising solutions involve resorting to SAT/SMT solvers to optimize the search phase of the reasoner [12] or to use well-established techniques from the field of Integer Linear Programming (ILP) [7] in order to generate solutions to the given satisfiability problem [14].

A prominent technique used to solve problems in ILP is *column generation* [6], a method for solving large integer linear systems by restricting the focus on a small subset of the columns of the original coefficient matrix and incrementally adding columns by means of an oracle, halting when an optimal solution is found. This technique has been successfully applied to reasoning services for several logics; among these logics is the first-order fragment of *counting quantifiers over unary predicates* [11], where satisfiability of a formula has been shown to be equivalent to find an integral and non-negative solution to a linear system of inequalities over a 0/1 coefficient matrix.

We reference [2] and [11] as the works that sparked interest in the topics developed in this thesis; most of the references that are cited throughout this work originate from these publications. Starting from the DL $\mathcal{ALCSCC}$ defined in [2] and the application of column generation technique to the problem of CQU formula satisfiability illustrated in [11], the goal of this thesis is to come up with an algorithm to decide concept satisfiability in a variant of $\mathcal{ALCSCC}$, replacing the theoretically correct but practically inefficient algorithm given in [2] with a decision procedure that is efficient for practical purposes, employing the column generation technique.

OUTLINE. In Chapter 2 we define some elementary notions and terminology, mainly related to *Description Logics* and *Integer Linear Programming*, to establish a background to develop upon in the next chapters. In particular, we take a look at two existing DLs, $\mathcal{ALCQ}$ and $\mathcal{ALCSCC}$, that enable quantitative reasoning over role successors and enunciate some of their relevant properties. For the latter DL, we introduce a variant called $\mathcal{ALCSCC}^{\infty}$ that allows reasoning over infinite sets.

In Chapter 3 we introduce a new DL, called $\mathcal{ALCCQU}$, initially defined as a syntactic restriction of $\mathcal{ALCSCC}^{\infty}$. Throughout the chapter, we define a bisimulation that characterizes concept descriptions of a DL that is equivalent to $\mathcal{ALCCQU}$, called $\mathcal{ALCQt}$, to then provide a model-theoretic characterization of $\mathcal{ALCQt}$ as a first-order fragment that is invariant under the newly-defined bisimulation. After that, we classify the DLs $\mathcal{ALCQ}$, $\mathcal{ALCCQU}$ and $\mathcal{ALCSCC}^{\infty}$ according to their relative expressive power. Finally, we show that $\mathcal{ALCCQU}$ possesses a precise characterization as the first-order fragment of $\mathcal{ALCSCC}^{\infty}$, completing our inquiry into the theoretical properties of $\mathcal{ALCCQU}$.

In Chapter 4 we present the first-order fragment CQU [11] and we describe how the column generation technique is employed to solve a linear system of inequalities. After that, we propose an algorithm for $\mathcal{ALCCQU}$ concept satisfiability that combines column generation with the branch-and-bound method for linear programming, adapting the approach taken in [11]. We show that

the algorithm that we propose is correct and terminating. We briefly conclude by delving into complexity-related considerations for the designed decision procedure.

2 Preliminaries

Before defining the logic $\mathcal{ALCCQU}$, we provide some preliminary knowledge that is going to be used throughout Chapter 3 and Chapter 4. We begin by setting the terminology regarding first-order logic and ILP [7]. For first-order logic, we also mention the class of *ω-saturated* interpretations [8], subject of Section 3.3. After that, we provide the definition of the DL $\mathcal{ALCQ}$ [4], together with the notion of *counting bisimulation* [16] that can be used to prove properties about the expressive power of extensions of $\mathcal{ALCQ}$. Finally, we present the DL $\mathcal{ALCSCC}$ that strictly extends $\mathcal{ALCQ}$ with expressive role successor constraints [2] and we mention results related to the complexity of reasoning in this setting and the relative expressive power with respect to $\mathcal{ALCQ}$.

2.1 First-order logic

We assume a basic knowledge of the following concepts: *propositional formula*, *truth assignment*, *first-order formula*, *interpretation*, *variable assignment*, *model*, *satisfiability*, *tautology*. Otherwise, the reader is referred to any introductory textbook on propositional and first-order logic, such as [9]. In this section, we introduce definitions and results that are relevant in further sections and that are not considered as background knowledge.

If F is a propositional formula, we denote with μ a truth assignment and with $\mu \models F$ the fact that μ is a model of F. Similarly, φ stands for a first-order formula — we omit the variables, where unnecessary —, $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ for a first-order interpretation, $\{x_1/d_1, \ldots, x_n/d_n\}$ for the truth assignment that maps the variable x_i to $d_i \in \Delta^{\mathcal{I}}$ for $i = 1, \ldots, n$ and $\mathcal{I}, \{x/d\} \models \varphi(x)$ if $\mathcal{I}$ is a model of $\varphi(x)$ under the truth assignment $\{x/d\}$.

In Section 3.3 we prove that the newly-introduced DL $\mathcal{ALCCQU}$ defined in Chapter 3 has a precise characterization as a fragment of first-order logic. To do so, we rely on some fundamental results, mentioned here.

Theorem 1 (Compactness theorem)**.** *A set of first-order formulae Γ is satisfiable if and only if every finite subset of Γ is satisfiable.*

Definition 1. An interpretation $\mathcal{I}$ is *ω-saturated* if for every set of first-order formulae Γ where only finitely many individuals from $\Delta^{\mathcal{I}}$ appear as constants,

$$\text{if every finite subset of } \Gamma \text{ is realizable, then } \Gamma \text{ is realizable.}$$

Theorem 2 ([8])**.** *For every interpretation $\mathcal{I}$ there exists an interpretation $\mathcal{I}'$ that is ω-saturated and satisfies the same first-order sentences as $\mathcal{I}$.*

2.2 Linear Programming

The branch of mathematical optimization called *Integer Linear Programming* (ILP) [7] studies methods and algorithms that are used to find optimal solutions to problems in linear algebra. Here, we set the basic terminology used in Chapter 4 in the development of an algorithm to decide $\mathcal{ALCCQU}$ concept satisfiability.

The linear programming problem. Given a *cost* vector $\mathbf{c} = (c_1, \ldots, c_n)$, a sequence $\mathbf{x} = (x_1, \ldots, x_n)$ of *decision variables* and a set of linear inequalities over $x_1, \ldots, x_n$ represented by the linear system $A\mathbf{x} \bowtie \mathbf{b}$ with $\mathbf{b} = (b_1, \ldots, b_m)$ and A a $m \times n$ matrix, the associated linear programming problem — also called *primal* problem — is

$$\begin{aligned} &\text{minimize } \mathbf{c}' \cdot \mathbf{x} \\ &\text{subject to } A \cdot \mathbf{x} \bowtie \mathbf{b} \text{ and } \mathbf{x} \geq \mathbf{0} \end{aligned} \tag{2.1}$$

where $\mathbf{c}'\mathbf{x} = \sum_{i=1}^{n} c_i x_i$ is the *cost function* of the problem. If the vector $\mathbf{x}$ satisfies all the constraints of the problem it is called a *feasible solution*; the set of all feasible solutions is called *feasible set*. A feasible solution that minimizes the cost function is called a *optimal solution*. Given the vector $\bowtie$ of m inequalities where $\bowtie_i \in \{\leq, \geq\}$ for $i = 1, \ldots, m$, we denote with $\bowtie^{-1}$ the vector obtained by switching the inequalities contained in $\bowtie$.

The *dual* problem associated to A, $\mathbf{b}$ and $\mathbf{c}$ is

$$\begin{aligned} &\text{maximize } \mathbf{z}' \cdot \mathbf{b} \\ &\text{subject to } \mathbf{z}' \cdot A \leq \mathbf{c}' \text{ and } \mathbf{z} \bowtie^{-1} \mathbf{0} \end{aligned} \tag{2.2}$$

where $\mathbf{z} = (z_1, \ldots, z_m)$ is the vector of *price variables*; a vector $\mathbf{z}$ satisfying the constraints of (2.2) is called a*dual solution* of (2.1).

If we additionally require that $\mathbf{x}$ is a vector of integers, (2.1) becomes an instance of an *Integer Linear Programming* (ILP) problem. Given an ILP problem, a vector $\mathbf{x}$ is a *relaxed* solution if it satisfies all the constraints but some of its values are non-integral.

Solving a linear program. There are many well-established techniques to solve integer linear programs, in both settings where feasibility or optimality of a solution are required. In Section 4.2 we take a look at *column generation* [6], a method used in solving integer linear programs that focuses only on a small subset of the problem, in order to generate optimal solutions. The explanation of other techniques, such as the *simplex method*, can be found in [7].

2.3 The description logic $\mathcal{ALCQ}$

The description logic $\mathcal{ALCQ}$ [4] is an extension of the well-known DL $\mathcal{ALC}$ with *qualified number restrictions*, that allow to state basic quantitative knowledge about the role successors of a given individual.

Example 1. In $\mathcal{ALCQ}$, it is possible to state that an individual that is a parent has at least two daughters and at most one son. The syntactic expression that defines such a concept is

$$\texttt{Parent} \sqcap (\geq 2\ \texttt{hasChild.}\ \texttt{Female}) \sqcap (\leq 1\ \texttt{hasChild.}\ \texttt{Male}).$$

We briefly introduce the syntax of $\mathcal{ALCQ}$ together with its semantics. After that, we cite a result from [16] stating that two individuals that can be related by means of a binary relation called *counting bisimulation* [16] are equivalent under $\mathcal{ALCQ}$, that is, they are described by the same concepts. We can use this equivalence relation to relate the expressive power of $\mathcal{ALCQ}$ to that of other DLs.

Syntax and semantics of $\mathcal{ALCQ}$. Given disjoint finite sets N_C and N_R of concept names and role names, respectively, the set of $\mathcal{ALCQ}$ concept descriptions is defined inductively:

- Every concept name in N_C is a $\mathcal{ALCQ}$ concept description;
- If C, D are $\mathcal{ALCQ}$ concept descriptions, r is a role name in N_R and $n \geq 0$ is a natural number, then $\neg C$ (negation), $C \sqcup D$ (disjunction), $C \sqcap D$ (conjunction), $(\geq\ n\ r.\ C)$ and $(\leq n\ r.\ C)$ (qualified number restrictions) are $\mathcal{ALCQ}$ concept descriptions.

We define the semantics of $\mathcal{ALCQ}$ using the notion of an interpretation. An *interpretation* $\mathcal{I}$ consists of a non-empty set $\Delta^{\mathcal{I}}$ called *domain* and a function $\cdot^{\mathcal{I}}$ that maps every concept name A to a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ and every role name r to a binary relation $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. This function is then extended to $\mathcal{ALCQ}$ concept descriptions as follows:

- $(\neg C)^{\mathcal{I}} := \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$, $(C \sqcup D)^{\mathcal{I}} := C^{\mathcal{I}} \cup D^{\mathcal{I}}$ and $(C \sqcap D)^{\mathcal{I}} := C^{\mathcal{I}} \cap D^{\mathcal{I}}$;
- $(\geq n\ r.\ C)^{\mathcal{I}} := \{x \in \Delta^{\mathcal{I}} \mid \big|\{y \in \Delta^{\mathcal{I}} \mid (x,y) \in r^{\mathcal{I}}\}\big| \geq n\}$
- $(\leq n\ r.\ C)^{\mathcal{I}} := \{x \in \Delta^{\mathcal{I}} \mid \big|\{y \in \Delta^{\mathcal{I}} \mid (x,y) \in r^{\mathcal{I}}\}\big| \leq n\}$

Given a $\mathcal{ALCQ}$ concept description C, we say that C is *satisfiable* if there exists an interpretation $\mathcal{I}$ such that $C^{\mathcal{I}} \neq \emptyset$ — notation $\mathcal{I} \models C$. Two $\mathcal{ALCQ}$ concept descriptions C and D are *equivalent* — notation $C \equiv D$ — if $C^{\mathcal{I}} = D^{\mathcal{I}}$ holds for every interpretation $\mathcal{I}$.

Counting bisimulation. As mentioned before, it is possible to state results about the expressive power of $\mathcal{ALCQ}$ with respect to another description logic by means of *counting bisimulation* [16]. In particular, we are going to use the results of this paragraph to show that the DL $\mathcal{ALCCQU}$, introduced in Chapter 3, is more expressive than $\mathcal{ALCQ}$, in Section 3.4.

Definition 2. Let $\mathcal{I}_1$ and $\mathcal{I}_2$ be interpretations. The relation $\rho \subseteq \Delta^{\mathcal{I}_1} \times \Delta^{\mathcal{I}_2}$ is a *counting bisimulation* between $\mathcal{I}_1$ and $\mathcal{I}_2$ if

1. $d_1\ \rho\ d_2$ implies

$$d_1 \in A^{\mathcal{I}_1} \text{ if and only if } d_2 \in A^{\mathcal{I}_2}$$

for all $d_1 \in \Delta^{\mathcal{I}_1}$, $d_2 \in \Delta^{\mathcal{I}_2}$ and $A \in N_C$.

2. if $d_1 \ \rho \ d_2$ and $D_1 \subseteq r^{\mathcal{I}_1}(d_1)$ is finite for $r \in N_R$, then there is a set $D_2 \subseteq r^{\mathcal{I}_2}(d_2)$ such that ρ contains a bijection between D_1 and D_2.

3. if $d_1 \ \rho \ d_2$ and $D_2 \subseteq r^{\mathcal{I}_2}(d_2)$ is finite for $r \in N_R$, then there is a set $D_1 \subseteq r^{\mathcal{I}_1}(d_1)$ such that ρ contains a bijection between D_1 and D_2.

The individuals $d_1 \in \Delta^{\mathcal{I}_1}$ and $d_2 \in \Delta^{\mathcal{I}_2}$ are *$\mathcal{ALCQ}$-bisimilar* — notation $(\mathcal{I}_1, d_1) \sim_{\mathcal{ALCQ}} (\mathcal{I}_2, d_2)$ — if there is a counting bisimulation ρ between $\mathcal{I}_1$ and $\mathcal{I}_2$ such that $d_1 \ \rho \ d_2$ and *$\mathcal{ALCQ}$-equivalent* — notation $(\mathcal{I}_1, d_1) \equiv_{\mathcal{ALCQ}} (\mathcal{I}_2, d_2)$ — if for all $\mathcal{ALCQ}$ concept descriptions C, $d_1 \in C^{\mathcal{I}_1}$ if and only if $d_2 \in C^{\mathcal{I}_2}$.

Theorem 3. *If* $(\mathcal{I}_1, d_1) \sim_{\mathcal{ALCQ}} (\mathcal{I}_2, d_2)$ *then* $(\mathcal{I}_1, d_1) \equiv_{\mathcal{ALCQ}} (\mathcal{I}_2, d_2)$.

Proof. Omitted. For details, refer to [16]. □

2.4 Extending $\mathcal{ALCQ}$ with expressive role successor constraints

The DL $\mathcal{ALCSCC}$ has been first presented in [2] as an extension of $\mathcal{ALCQ}$ that allows to express constraints over role successors of an individual using formulae in the logic fragment QFBAPA [15] with structural restrictions. In this section, we introduce a semantic variant of QFBAPA, called QFBAPA$^\infty$, where solutions need not to be finite. After that, we define the DL $\mathcal{ALCSCC}^\infty$, which is a variant of $\mathcal{ALCSCC}$ that allows for constraints expressible in QFBAPA$^\infty$ and observe what results that hold in $\mathcal{ALCSCC}$, shown in [2], might not necessarily be valid in the context of $\mathcal{ALCSCC}^\infty$.

2.4.1 The logic QFBAPA$^\infty$

The logic QFBAPA$^\infty$, similarly to its well-known variant QFBAPA [15], allows one to build *set terms* as Boolean combinations over a finite set of symbols and to impose set and cardinality constraints expressed in Presburger arithmetics over these terms.

Set and cardinality terms. Given a finite set of symbols T with $\{\emptyset, \mathcal{U}\} \cap T = \emptyset$, the *set terms* over T are inductively defined as follows:

1. $\emptyset$ (empty set) and $\mathcal{U}$ (universe) are set terms over T;

2. all the symbols in T are set terms over T;

3. if s, t are set terms over T then $s \cup t$ (union), $s \cap t$ (intersection) and s^c (complement) are set terms over T.

The *cardinality terms* (or *Presburger expressions*) over T are inductively defined as follows:

1. every element of $\mathbb{N}$ is a cardinality term;

2. if s is a set term over T then $|s|$ is a cardinality term over T;

3. if k, ℓ are cardinality terms over T then $k + \ell$ (sum) and $N \cdot \ell$ (multiplication by constant) are cardinality terms over T, with $N \in \mathbb{N}$ a natural number.

Constraints and solutions. *Set constraints* over T are assertions of the form $s \subseteq t, s = t$ or their negation for set terms s, t over T. *Cardinality constraints* over T are assertions of the form $k = \ell, k > \ell, k \geq \ell$ and their negation or $N \mathtt{dvd} k$ — which expresses the fact that k is a multiple of N — for cardinality terms k, ℓ and N a natural number. A QFBAPA$^\infty$ formula consists of a Boolean combination of set and cardinality constraints.

An interpretation $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ over T consists of a non-empty domain set $\Delta^\mathcal{I}$ and a mapping $\cdot^\mathcal{I}$ that assigns the set $\emptyset^\mathcal{I} = \emptyset$ to the symbol $\emptyset$, a finite set $\mathcal{U}^\mathcal{I} \subseteq \Delta^\mathcal{I}$ to the symbol $\mathcal{U}$ and a subset $\sigma^\mathcal{I} \subseteq \mathcal{U}^\mathcal{I}$ to each set symbol σ in T. We extend this mapping to set terms and cardinality terms as follows:

1. $(s \cup t)^\mathcal{I} = s^\mathcal{I} \cup t^\mathcal{I}$, $(s \cap t)^\mathcal{I} = s^\mathcal{I} \cap t^\mathcal{I}$ and $(s^c)^\mathcal{I} = \mathcal{U}^\mathcal{I} \setminus s^\mathcal{I}$,

2. $|s|^\mathcal{I} = \left|s^\mathcal{I}\right|$,

3. $(k + \ell)^\mathcal{I} = k^\mathcal{I} + \ell^\mathcal{I}$ and $(N \cdot \ell)^\mathcal{I} = N \cdot \ell^\mathcal{I}$.

The mapping $\cdot^\mathcal{I}$ satisfies:

1. the set constraint $s \subseteq t$ if $s^\mathcal{I} \subseteq t^\mathcal{I}$ and its negation if $s^\mathcal{I} \not\subseteq t^\mathcal{I}$;

2. the set constraint $s = t$ if $s^\mathcal{I} = t^\mathcal{I}$ and its negation if $s^\mathcal{I} \neq t^\mathcal{I}$;

3. the cardinality constraint $k \leq \ell$ if $k^\mathcal{I} \leq \ell^\mathcal{I}$ and its negation if $k^\mathcal{I} > \ell^\mathcal{I}$;

4. the cardinality constraint $k < \ell$ if $k^\mathcal{I} < \ell^\mathcal{I}$ and its negation if $k^\mathcal{I} \geq \ell^\mathcal{I}$;

5. the cardinality constraint $k = \ell$ if $k^\mathcal{I} = \ell^\mathcal{I}$ and its negation if $k^\mathcal{I} \neq \ell^\mathcal{I}$;

6. the divisibility constraint $N \mathtt{dvd} k$ if there exists a natural number $M \geq 0$ such that $k^\mathcal{I} = N \cdot M$.

The interpretation $\mathcal{I}$ satisfies the QFBAPA$^\infty$ formula φ if it satisfies the Boolean combination of set and cardinality constraints contained in φ.

The main difference between QFBAPA$^\infty$ and the definition of QFBAPA provided in [2] is that QFBAPA$^\infty$ set terms are not necessarily interpreted as finite sets. In particular, in QFBAPA$^\infty$ it is possible to enforce infinity of a set in every solution. For instance, the every solution that satisfies $|\mathcal{U}| = |\mathcal{U}| + 1$ maps $\mathcal{U}$ to an infinite set.

As we are going to show in the next subsection, in $\mathcal{ALCSCC}^\infty$ one can express constraints over role successors as a finite conjunction of QFBAPA$^\infty$ set and cardinality constraints over a set of symbols related to role and concept names.

2.4.2 The description logic $\mathcal{ALCSCC}^\infty$

The DL $\mathcal{ALCSCC}^\infty$ that we define in this section presents some non-trivial differences with respect to the DL $\mathcal{ALCSCC}$ introduced in [2]. In $\mathcal{ALCSCC}$, divisibility constraints can be used to form role successor constraints, whereas $\mathcal{ALCSCC}^\infty$ restricts the admissible constraints to set and cardinality comparison ones. On the other hand, in $\mathcal{ALCSCC}^\infty$ sets can have an infinite interpretation. In a later paragraph, we show how divisibility constraints in $\mathcal{ALCSCC}^\infty$ would allow to specify the very strong property that an individual has only finitely many role successors. Finally, we lift the assumption that every individual in the domain of an interpretation must have only finitely many successors.

Syntax of $\mathcal{ALCSCC}^\infty$. Given a countable set N_C of concept names and a *finite* set N_R of role names that are disjoint, the set of $\mathcal{ALCSCC}^\infty$ concept descriptions over the signature (N_C, N_R) is inductively defined as follows:

1. every concept name in N_C is an $\mathcal{ALCSCC}^\infty$ concept description;
2. if C, D are $\mathcal{ALCSCC}^\infty$ concept descriptions, then so are $C \sqcap D$, $C \sqcup D$ and $\neg C$;
3. if c is a set or cardinality constraint, different from a divisibility constraint, over a finite set of symbols containing role names and $\mathcal{ALCSCC}^\infty$ concept descriptions, then $\mathrm{succ}(c)$ is an $\mathcal{ALCSCC}^\infty$ concept description.

The symbols $\top$ (top) and $\bot$ (bottom) are introduced as abbreviations for $A \sqcup \neg A$ and $A \sqcap \neg A$, with $A \in N_C$.

Example 2. In $\mathcal{ALCSCC}^\infty$ we can describe an individual that is a parent and has the same number of sons and daughters, by using the concept description

$$C := \texttt{Parent} \sqcap \mathrm{succ}(|\texttt{hasChild} \cap \texttt{Male}| = |\texttt{hasChild} \cap \texttt{Female}|).$$

Differently from the definition of $\mathcal{ALCSCC}$ given in [2], we do not assume that the set of concept names is finite. However, we still require that the set of role names N_R is finite, in order to be able to specify the semantics of $\mathcal{ALCSCC}^\infty$ as intended.

Natural numbers and infinity. We are not interested in distinguishing cardinalities of infinite sets from one another. Therefore, we consider the set $\mathbb{N}_\infty := \mathbb{N} \cup \{\infty\}$ of natural numbers extended with the *infinity* ∞, which satisfies the following axiom schemata, for all $n \in \mathbb{N}$:

$$\begin{aligned} n + \infty = \infty + n = \infty, \qquad & n \cdot \infty = \infty \cdot n = \infty, \\ \infty + \infty = \infty \cdot \infty = \infty, \qquad & n < \infty. \end{aligned}$$

Hereafter, we assume that the cardinality mapping $|\cdot|$ maps set terms to values in $\mathbb{N}_\infty$.

SEMANTICS OF $\mathcal{ALCSCC}^\infty$. An interpretation $\mathcal{I}$ of N_C and N_R consist of a non-empty set $\Delta^\mathcal{I}$called *domain* and a mapping $\cdot^\mathcal{I}$ that maps every concept name $A \in N_C$ to a set $A^\mathcal{I} \subseteq \Delta^\mathcal{I}$ and every role name $r \in N_R$ to a binary relation $r^\mathcal{I}$ over $\Delta^\mathcal{I}$. For a given individual $d \in \Delta^\mathcal{I}$, we denote the set of its r-successors in $\Delta^\mathcal{I}$ by $r^\mathcal{I}(d)$.

The mapping $\cdot^\mathcal{I}$ is inductively extended to $\mathcal{ALCSCC}^\infty$ concept descriptions as follows:

- $(C \sqcap D)^\mathcal{I} := C^\mathcal{I} \cap D^\mathcal{I}$, $(C \sqcup D)^\mathcal{I} := C^\mathcal{I} \cup D^\mathcal{I}$ and $(\neg C)^\mathcal{I} := \Delta^\mathcal{I} \setminus C^\mathcal{I}$;
- $\mathrm{succ}(c)^\mathcal{I} := \{d \in \Delta^\mathcal{I} \mid \mathcal{I}_d \text{ satisfies } c\}$, where $\mathcal{I}_d$ is an interpretation over set terms such that

$$\emptyset^{\mathcal{I}_d} := \emptyset, \quad \Delta^{\mathcal{I}_d} = \mathcal{U}^{\mathcal{I}_d} := \bigcup_{r \in N_R} r^\mathcal{I}(d), \quad C^{\mathcal{I}_d} := C^\mathcal{I} \cap \mathcal{U}^{\mathcal{I}_d}, \quad r^{\mathcal{I}_d} := r^\mathcal{I}(d)$$

  for all $\mathcal{ALCSCC}^\infty$ concept descriptions C and role names r occurring in c.

The interpretation $\mathcal{I}_d$ is well-defined: indeed, we can assume by induction that $C^\mathcal{I}$ is already defined for every concept description C occurring in c. For every interpretation $\mathcal{I}$, it holds that $\top^\mathcal{I} = \Delta^\mathcal{I}$ and $\bot^\mathcal{I} = \emptyset^\mathcal{I}$. Differently from [2], the set $\mathcal{U}^{\mathcal{I}_d}$ is not guaranteed to be finite.

DIVISIBILITY CONSTRAINT AND INFINITY. The definition of the DL $\mathcal{ALCSCC}$ in [2] allows to specify divisibility successor constraints of the form

$$\mathrm{succ}(N \ \mathtt{dvd}\ k)$$

where N is a natural number and k is a cardinality term; such a term is satisfied by $\mathcal{I}$ if $k^\mathcal{I}$ is divisible by N. The extension of natural numbers with infinity poses a substantial issue: is the infinity odd, even, both or neither? We can surely exclude that ∞ has a specific parity: if ∞ was even (resp. odd), then $\infty + 1$ would be odd (resp. even), but $\infty = \infty + 1$, therefore ∞ would also be odd (resp. even). If ∞ was assumed to be both odd and even, we would be able to state that a concept description has only finitely many role successors using

$$\mathtt{finite} := \neg\,\mathrm{succ}(2 \ \mathtt{dvd}\ |\mathcal{U}|) \sqcup \neg\,\mathrm{succ}(3 \ \mathtt{dvd}\ |\mathcal{U}|)$$

as an additional conjunct. Similarly, if ∞ was assumed to be neither odd nor even, the concept description

$$\mathtt{finite} := \mathrm{succ}(2 \ \mathtt{dvd}\ |\mathcal{U}|) \sqcup \mathrm{succ}(3 \ \mathtt{dvd}\ |\mathcal{U}|)$$

would enforce every individual in every interpretation of `finite` to have only finitely many role successors.

EXPRESSIVE POWER. The ability to state constraints over role successors using QFBAPA$^\infty$ formulae strictly increases the expressive power of $\mathcal{ALCSCC}^\infty$ with respect to $\mathcal{ALCQ}$. This is already proved to be true for $\mathcal{ALCSCC}$ in [2], showing that the concept description $\mathrm{succ}(|r| = |s|)$ has no equivalent $\mathcal{ALCQ}$ concept. In Section 3.4, we provide a proof based on the use of counting bisimulation to show that both the DLs $\mathcal{ALCSCC}^\infty$ and $\mathcal{ALCCQU}$ (defined in Chapter 3) are strictly more expressive than $\mathcal{ALCQ}$.

CONCEPT SATISFIABILITY IN $\mathcal{ALCSCC}^\infty$ The complexity of the concept satisfiability problem in $\mathcal{ALCSCC}$ without a TBox has been exactly determined in [2].

Theorem 4. *The problem of checking satisfiability of $\mathcal{ALCSCC}$ concept satisfiability is PSPACE-complete.*

Proof. Omitted. For details, refer to [2]. □

The proof given in [2] relies on the assumption that the set $\mathcal{U}^{\mathcal{I}}$ is finite for every interpretation $\mathcal{I}$ of a QFBAPA formula; this condition is met in the original definition of $\mathcal{ALCSCC}$. Under this hypothesis, one can leverage the following result about the *Venn regions* over the variables of a QFBAPA formula φ — if $X_1, \ldots, X_k$ are the variables occurring in φ, a Venn region is a set term of the form $X_1^{p_1} \cap \cdots \cap X_k^{p_k}$, where $X_i^{p_i}$ is either X_i or X_i^c.

Lemma 1. *For every QFBAPA formula φ, one can compute in polynomial time a number N whose value is polynomial in the size of φ such that if σ is a solution of φ, then there exists a solution σ' of φ satisfying*

$$\begin{aligned}&\big|\{v \mid v \textit{ Venn region}, \sigma'(v) \neq 0\}\big| \leq N,\\ &\{v \mid v \textit{ Venn region}, \sigma'(v) \neq 0\} \subseteq \{v \mid v \textit{ Venn region}, \sigma(v) \neq 0\}.\end{aligned}$$

The condition of applicability of Lemma 1 is not met in $\mathcal{ALCSCC}^\infty$: the universe set $\mathcal{U}^{\mathcal{I}_d}$ of role successors of an individual d under an interpretation $\mathcal{I}$ is not required to be finite, whereas the semantics of QFBAPA requires that the universe is mapped to a finite set under every interpretation. We conjecture, however, that Theorem 4 and Lemma 1 ought to be valid for $\mathcal{ALCSCC}^\infty$ as well.

PRACTICAL REASONING IN $\mathcal{ALCSCC}^\infty$. While the algorithm proposed in [2] establishes important complexity results and provides a worst-case optimal decision procedure for $\mathcal{ALCSCC}$ concept satisfiability without a knowledge base, it lacks the practical efficiency that is desirable in order to obtain a usable implementation. In particular, the algorithm relies on non-deterministic operations, such as guessing a truth assignment and guessing the polynomial number of Venn regions mentioned in Lemma 1.

In the next chapter, we restrict our attention to a sublogic of $\mathcal{ALCSCC}^\infty$, called $\mathcal{ALCCQU}$, where the only cardinality constraints that can be used in role successor constraints are those of the form $k \leq N$, $k = N$ and $k \geq N$ (or their negation) with k a complex cardinality term and $N \geq 0$ a natural number. We develop a model-theoretic characterization of $\mathcal{ALCCQU}$ as a specific fragment of first-order logic and we show that its expressive power is placed between $\mathcal{ALCQ}$ and $\mathcal{ALCSCC}^\infty$. Finally, we show that $\mathcal{ALCCQU}$ can be characterized as a particular fragment of $\mathcal{ALCSCC}^\infty$, namely, the fragment of $\mathcal{ALCSCC}^\infty$ that is within first-order logic.

3 The description logic $\mathcal{ALCCQU}$

In this chapter, we focus our attention to a restriction of $\mathcal{ALCSCC}^{\infty}$ called $\mathcal{ALCCQU}$. As explained at the end of Chapter 2, the DL $\mathcal{ALCCQU}$ is obtained by restricting the form of the admissible role successor constraints. Throughout the chapter, we are going to show that such a restriction provides some interesting properties; the most interesting one is that $\mathcal{ALCCQU}$ can be embedded into first-order logic using an appropriate translation, whereas there are $\mathcal{ALCSCC}^{\infty}$ concept descriptions that are beyond first-order logic. At the end of this chapter, we show that $\mathcal{ALCCQU}$ has an exact characterization as the first-order fragment of $\mathcal{ALCSCC}^{\infty}$.

Syntax and semantics of $\mathcal{ALCCQU}$. Given a countable set N_C of concept names and a *finite* set N_R of role names that are disjoint, the set of $\mathcal{ALCCQU}$ concept descriptions over the signature (N_C, N_R) is inductively defined as follows:

1. every concept name in N_C is an $\mathcal{ALCCQU}$ concept description;
2. if C, D are $\mathcal{ALCCQU}$ concept descriptions, then so are $C \sqcap D$ (conjunction), $C \sqcup D$ (disjunction) and $\neg C$ (negation);
3. if c is a set constraint or a cardinality constraint of the form $k = N$, $k > N$ or $k \geq N$ or their negation, with k a cardinality term over a finite set of symbols containing role names and $\mathcal{ALCCQU}$ concept descriptions and $N \in \mathbb{N}$, then $\mathrm{succ}(c)$ (role successor constraint) is an $\mathcal{ALCCQU}$ concept description.

Hereafter, where the context makes it clear, we refer to the set terms defined over role names and $\mathcal{ALCCQU}$ concept descriptions simply as *set terms*.

Example 3. In $\mathcal{ALCCQU}$ we can describe an individual that is a parent and whose children are not living with their parents, using the concept description

$$C := \mathrm{succ}(|\texttt{hasChild} \cap \texttt{livesWith}| = 0).$$

In Section 3.4 we show (using different role names) that this concept description cannot be expressed in $\mathcal{ALCQ}$.

The semantics of $\mathcal{ALCCQU}$ in terms of an interpretation $\mathcal{I}$ is defined in the same way as the semantics of $\mathcal{ALCSCC}$ in Section 2.4, restricted to the cardinality constraints appearing in $\mathcal{ALCCQU}$.

3.1 A normal form for $\mathcal{ALCCQU}$

In this section, we define a method to reduce $\mathcal{ALCCQU}$ role successor constraints to a specific syntactic form, that is, we prove that every set and cardinality constraint can be equivalently expressed

using only cardinality constraints of the form $|s| \geq N$ or $|s| \leq N$ where s is a set term and N a natural number. This transformation can be used to restrict our algorithm for $\mathcal{ALCCQU}$ concept satisfiability, introduced in Chapter 4, to only deal with role successor constraints containing cardinality constraints of the form mentioned above. Moreover, we can push the negation symbols in the input concept description so that they only appear in front of atomic concept names. If C is the concept description of interest, the concept $\mathtt{nf}(C)$ obtained by applying exhaustively these transformations is called its *$\mathcal{ALCCQU}$-normal form*.

Definition 3. A $\mathcal{ALCCQU}$ concept description C is in *$\mathcal{ALCCQU}$-normal form* if negation symbols only appear in front of concept names occurring in C and each role successor constraint occurring in C contains a cardinality constraint of the form $|s| \leq N$ or $|s| \geq N$ with N a natural number and s a set term.

Bringing $\mathcal{ALCCQU}$ concepts to normal form. It is possible to replace every $\mathcal{ALCCQU}$ role successor constraint with a restricted form, as suggested at the beginning of this section.

Set constraints can be replaced by cardinality constraints using the rules induced by the following equivalences:

$$\mathrm{succ}(s \subseteq t) \equiv \mathrm{succ}(|s \cap t^c| = 0), \tag{3.1}$$
$$\mathrm{succ}(s = t) \equiv \mathrm{succ}(s \subseteq t) \sqcap \mathrm{succ}(t \subseteq s). \tag{3.2}$$

We adopt the following replacement rules to convert general cardinality constraints to expressions of the form $k \geq N$ or $k \leq N$:

$$\mathrm{succ}(k > N) \rightsquigarrow \mathrm{succ}(k \geq N + 1) \tag{3.3}$$
$$\mathrm{succ}(k < N) \rightsquigarrow \mathrm{succ}(k \leq N - 1) \tag{3.4}$$
$$\mathrm{succ}(k = N) \rightsquigarrow \mathrm{succ}(k \geq N) \sqcap \mathrm{succ}(k \leq N) \tag{3.5}$$
$$\mathrm{succ}(k \neq N) \rightsquigarrow \mathrm{succ}(k \geq N + 1) \sqcup \mathrm{succ}(k \leq N - 1) \tag{3.6}$$
$$\mathrm{succ}(\textstyle\sum_{i=1}^{M} k_i \bowtie N) \rightsquigarrow \bigsqcup\{\prod_{i=1}^{M} \mathrm{succ}(k_i \bowtie N_i) \mid \sum_{i=1}^{M} N_i = N\} \tag{3.7}$$
$$\mathrm{succ}(M \cdot k \bowtie N) \rightsquigarrow \mathrm{succ}(\textstyle\sum_{i=1}^{M} k \bowtie N) \text{ with } \bowtie \in \{\leq, \geq\} \tag{3.8}$$

Finally, we employ the following rules to ensure that the negation symbol only appears in front of concept names:

$$\neg(C \sqcap D) \rightsquigarrow \neg C \sqcup \neg D, \quad \neg(C \sqcup D) \rightsquigarrow \neg C \sqcap \neg D, \quad \neg\neg C \rightsquigarrow C \tag{3.9}$$
$$\neg\,\mathrm{succ}(k \leq N) \rightsquigarrow \mathrm{succ}(k \geq N + 1), \quad \neg\,\mathrm{succ}(k \geq N) \rightsquigarrow \mathrm{succ}(k \leq N - 1) \tag{3.10}$$
$$\mathrm{succ}(k \leq -1) \rightsquigarrow \bot \tag{3.11}$$

The process where we first apply the rules induced by (3.1)– (3.2) to get rid of set constraints and then we exhaustively apply rules (3.3)– (3.11) is guaranteed to terminate: the nesting level of role successor constraints within one another in the input concept description C is finite and every rule generates at most exponentially many new role successor constraints where the nesting level is decreased.

Theorem 5. *For every $\mathcal{ALCCQU}$ concept description C there exists a $\mathcal{ALCCQU}$-normal form $\mathtt{nf}(C)$ obtained from C in exponential time such that $C \equiv \mathtt{nf}(C)$.*

Proof. We show that the application of rule (3.3) yields an equivalent concept description; the cases (3.4)–(3.6) and (3.10) can be proved analogously. For every interpretation $\mathcal{I}$,

$$\begin{aligned}\mathrm{succ}(k > N)^{\mathcal{I}} &= \{d \in \Delta^{\mathcal{I}} \mid k^{\mathcal{I}} > N\} \\ &= \{d \in \Delta^{\mathcal{I}} \mid k^{\mathcal{I}} \geq N+1\} = \mathrm{succ}(k \geq N+1)^{\mathcal{I}}.\end{aligned}$$

To prove that (3.7) yields an equivalent concept, we notice that the inequality $a_1 + \cdots + a_M \geq N$ is equivalent to the set of inequalities

$$a_i \geq N_i,\ i = 1, \ldots, M \wedge N_1 + \cdots + N_M = N.$$

The number of possible assignments of natural values to $N_1, \ldots, N_M$ such that $\sum_{i=1}^{M} N_i = N$ is equal to the number of partitions of N into exactly M parts, defined by the recurrence relation

$$P(N, M) = P(N-1, M-1) + P(N-M, M).$$

This means that each application of (3.7) produces $P(N, M) \cdot M$ new role successor constraints. The function $p_M(N) := P(N, M)$ has an asymptotic exponential growth [1], thus it generates at most exponentially many new role successor constraints. Finally, (3.8) naturally follows from the definition of arithmetic multiplication. The last equivalence for (3.11) follows from the fact that cardinality terms can only assume non-negative values. □

To conclude this section, we observe that (3.8) could be replaced with the rules

$$\mathrm{succ}(M \cdot k \geq N) \rightsquigarrow \mathrm{succ}(k \geq \left\lceil \frac{N}{M} \right\rceil)$$
$$\mathrm{succ}(M \cdot k \leq N) \rightsquigarrow \mathrm{succ}(k \leq \left\lfloor \frac{N}{M} \right\rfloor)$$

that still yield equivalent concept descriptions and avoid the exponential blowup caused by the conversion of multiplication into addition in (3.8).

Now that we have detailed the process to obtain a $\mathcal{ALCCQU}$-normal form, we proceed by looking for a characterization of $\mathcal{ALCCQU}$ that comes with a deeper nature than a simple syntactic restriction. In the next section, we show that $\mathcal{ALCCQU}$ is equivalent to another description logic that is obtained by generalizing $\mathcal{ALCQ}$ qualified number restrictions to include more expressive role expressions than just role names. We then investigate how this DL, called $\mathcal{ALCQ}t$, can be characterized as a specific fragment of first-order logic.

3.2 $\mathcal{ALCQ}t$ as an equivalent formulation of $\mathcal{ALCCQU}$

In this section, we define an extension of the DL $\mathcal{ALCQ}$ where qualified number restrictions are defined on *role types* instead of roles, called $\mathcal{ALCQ}t$. This DL can be seen as a special case

of the DL $\mathcal{ALCQI}b$ [20] where the following conditions hold: 1. inverse roles are not allowed, 2. disjunction is not allowed in role expressions, 3. the set N_R is finite. The DLs $\mathcal{ALCCQU}$ and $\mathcal{ALCQ}t$ turn out to be equivalent — as shown further in this section — although the encoding of $\mathcal{ALCCQU}$ concept descriptions into $\mathcal{ALCQ}t$ ones is a very expensive operation, thus ruling out the possibility of reducing $\mathcal{ALCCQU}$ concept satisfiability to $\mathcal{ALCQ}t$ concept satisfiability. On the other hand, this equivalence is handy because it allows to interchangeably use $\mathcal{ALCQ}t$ and $\mathcal{ALCCQU}$ to show properties regarding their expressive power with respect to other logics.

Syntax of $\mathcal{ALCQ}t$. Let N_C and N_R be respectively a countable set of *concept names* and a finite set of *role names* that are disjoint. A *role literal* over N_R is either a role name r or its negation $\neg r$. A *$\mathcal{ALCQ}t$-role type* ω over N_R is a finite conjunction of role literals where every element of N_R occurs exactly in one role literal. A role type over N_R is *safe* if it contains at least one positive literal. The set of $\mathcal{ALCQ}t$ concept descriptions over N_R and N_C is inductively defined as follows:

1. every concept name $A \in N_C$ is a $\mathcal{ALCQ}t$ concept description;
2. if C, D are $\mathcal{ALCQ}t$ concept descriptions, then so are $C \sqcap D$ (conjunction), $C \sqcup D$ (intersection) and $\neg C$ (negation);
3. if ω is a safe role type over N_R, C is a $\mathcal{ALCQ}t$ concept description and $N \geq 0$ is a natural number, then $(\geq\ N\ \omega.\ C)$ and $(\leq\ N\ \omega.\ C)$ (qualified number restrictions) are also $\mathcal{ALCQ}t$ concept descriptions.

Example 4. If $N_R = \{r, s, t\}$, then the expression $r \sqcap s \sqcap t$ is a safe role type, while $\neg r \sqcap \neg s \sqcap \neg t$ is not a safe role type. The expression $s \sqcap t$ is not a role type because r is not occurring in any literal role, while $r \sqcap \neg r \sqcap s \sqcap t$ is not a role type because r appears in more than one role literal.

Semantics of $\mathcal{ALCQ}t$. The semantics of $\mathcal{ALCQ}t$ in terms of an interpretation $\mathcal{I}$ is defined similarly to the semantics of $\mathcal{ALCQ}$ shown in Section 2.3. We extend that definition to include role types and qualified number restrictions over role types:

- $(\neg r)^{\mathcal{I}} := \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \setminus r^{\mathcal{I}}$ and $(\omega_1 \sqcap \omega_2)^{\mathcal{I}} := \omega_1^{\mathcal{I}} \cap \omega_2^{\mathcal{I}}$ with ω_1 and ω_2 role types;
- For qualified number restrictions over role types,

$$(\geq N\ \omega.\ C)^{\mathcal{I}} := \{d \in \Delta^{\mathcal{I}} \mid \big|\{(d, e) \in \omega^{\mathcal{I}} \mid e \in C^{\mathcal{I}}\}\big| \geq N\},$$
$$(\leq N\ \omega.\ C)^{\mathcal{I}} := \{d \in \Delta^{\mathcal{I}} \mid \big|\{(d, e) \in \omega^{\mathcal{I}} \mid e \in C^{\mathcal{I}}\}\big| \leq N\}.$$

The following lemma shows that the negation symbol can be absorbed by a qualified number restrictions.

Lemma 2. *Given an $\mathcal{ALCQ}t$ concept description C, a safe role type ω and $N \in \mathbb{N}$,*

$$(\leq N\ \omega.\ C) \equiv \neg(\geq N + 1\ \omega.\ C). \tag{3.12}$$

3.2.1 $\mathcal{ALCQ}t$ is a sublogic of $\mathcal{ALCCQU}$

The DL $\mathcal{ALCQ}t$ has a straightforward encoding in $\mathcal{ALCCQU}$ that is shown in the next lemma. This yields a translation of $\mathcal{ALCQ}t$ to $\mathcal{ALCCQU}$ in linear time with respect to the number of role successor constraints occurring in the input concept.

Lemma 3. *For each interpretation $\mathcal{I}$ and safe role ω,*

$$(\geq N\ \omega.\ C)^{\mathcal{I}} = \mathrm{succ}(\left|\omega \cap C^{\sharp}\right| \geq N)^{\mathcal{I}},$$
$$(\leq N\ \omega.\ C)^{\mathcal{I}} = \mathrm{succ}(\left|\omega \cap C^{\sharp}\right| \leq N)^{\mathcal{I}}$$

where $C^{\sharp}$ is the $\mathcal{ALCCQU}$ concept description equivalent to C.

Proof. Given $d \in \Delta^{\mathcal{I}}$, let $\omega^{\mathcal{I}}(d) := \{e \in \Delta^{\mathcal{I}} \mid (d,e) \in \omega^{\mathcal{I}}\}$. Then,

$$\begin{aligned}(\geq N\ \omega.\ C)^{\mathcal{I}} &= \{d \in \Delta^{\mathcal{I}} \mid \sharp\{(d,e) \in \omega^{\mathcal{I}} \mid e \in C^{\mathcal{I}}\} \geq N\} \\ &= \{d \in \Delta^{\mathcal{I}} \mid \left|\omega^{\mathcal{I}}(d) \cap C^{\mathcal{I}_d}\right| \geq N\}\end{aligned}$$

because ω is a safe role — thus $e \in \mathcal{U}^{\mathcal{I}_d}$ — and finally

$$= \{d \in \Delta^{\mathcal{I}} \mid |\omega \cap C|^{\mathcal{I}_d} \geq N\} = \mathrm{succ}(|\omega \cap C| \geq N)^{\mathcal{I}}.$$

The second identity can be proved analogously. □

3.2.2 $\mathcal{ALCCQU}$ is a sublogic of $\mathcal{ALCQ}t$

In Lemma 3 we showed that $\mathcal{ALCQ}t$ is a sublogic of $\mathcal{ALCCQU}$. The goal of this subsection is to prove that the converse holds as well, thus obtaining the equivalence of the two DLs.

The first result that we prove is that we can replace every set term that spans over $\mathcal{ALCCQU}$ concept descriptions with a unique $\mathcal{ALCCQU}$ concept, yielding an equivalent constraint.

Lemma 4. *Let $\mathcal{I}$ be an interpretation and $d \in \Delta^{\mathcal{I}}$ a fixed individual. For all $\mathcal{ALCCQU}$ concept descriptions C and D:*

1. $(\neg C)^{\mathcal{I}_d} = (C^c)^{\mathcal{I}_d}$,
2. $(C \sqcap D)^{\mathcal{I}_d} = (C \cap D)^{\mathcal{I}_d}$ *and* $(C \sqcup D)^{\mathcal{I}_d} = (C \cup D)^{\mathcal{I}_d}$
3. $\top^{\mathcal{I}_d} = \mathcal{U}^{\mathcal{I}_d}$ *and* $\bot^{\mathcal{I}_d} = \emptyset^{\mathcal{I}_d}$.

Proof. We first show the equality in (1), by using the semantics of $\mathcal{ALCCQU}$ concept descriptions for the interpretation $\mathcal{I}$ and the semantics of $\mathcal{ALCCQU}$ set terms for $\mathcal{I}_d$. Moreover, we use known equalities from set theory, such as the De Morgan laws and distributivity of intersection over union. We obtain the following chain of equalities, where (†) follows from the set-theoretic

identity $A \setminus (B \cap C) = (A \setminus B) \cap (A \setminus C)$ and (‡) from $A \cap (B \setminus C) = (A \cap B) \setminus C$ and $\mathcal{U}^{\mathcal{I}_d} \subseteq \Delta^{\mathcal{I}}$ (which also means that $\mathcal{U}^{\mathcal{I}_d} = \mathcal{U}^{\mathcal{I}_d} \cap \Delta^{\mathcal{I}}$).

$$\begin{aligned}(C^c)^{\mathcal{I}_d} &= \mathcal{U}^{\mathcal{I}_d} \setminus C^{\mathcal{I}_d} = \mathcal{U}^{\mathcal{I}_d} \setminus (\mathcal{U}^{\mathcal{I}_d} \cap C^{\mathcal{I}}) \overset{\dagger}{=} (\mathcal{U}^{\mathcal{I}_d} \setminus \mathcal{U}^{\mathcal{I}_d}) \cap (\mathcal{U}^{\mathcal{I}_d} \setminus C^{\mathcal{I}}) = \\ &= \mathcal{U}^{\mathcal{I}_d} \setminus C^{\mathcal{I}} = (\mathcal{U}^{\mathcal{I}_d} \cap \Delta^{\mathcal{I}}) \setminus C^{\mathcal{I}} \overset{\ddagger}{=} \mathcal{U}^{\mathcal{I}_d} \cap (\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}) = \\ &= \mathcal{U}^{\mathcal{I}_d} \cap (\neg C)^{\mathcal{I}} = (\neg C)^{\mathcal{I}_d}.\end{aligned}$$

The equalities in (2) are derived in a similar way; here, the equality (†) follows from the identity $\mathcal{U}^{\mathcal{I}_d} \cap \mathcal{U}^{\mathcal{I}_d} = \mathcal{U}^{\mathcal{I}_d}$. We only show the case for intersection, since the case for disjunction can be proved analogously:

$$\begin{aligned}(C \sqcap D)^{\mathcal{I}_d} &= \mathcal{U}^{\mathcal{I}_d} \cap (C \sqcap D)^{\mathcal{I}} = \mathcal{U}^{\mathcal{I}_d} \cap (C^{\mathcal{I}} \cap D^{\mathcal{I}}) = \\ &\overset{\dagger}{=} (\mathcal{U}^{\mathcal{I}_d} \cap C^{\mathcal{I}}) \cap (\mathcal{U}^{\mathcal{I}_d} \cap C^{\mathcal{I}}) = C^{\mathcal{I}_d} \cap D^{\mathcal{I}_d} = (C \cap D)^{\mathcal{I}_d}.\end{aligned}$$

The equalities in (3) are consequences of $\top^{\mathcal{I}_d} = \top^{\mathcal{I}} \cap \mathcal{U}^{\mathcal{I}_d} = \mathcal{U}^{\mathcal{I}_d}$ and $\bot^{\mathcal{I}_d} = \bot^{\mathcal{I}} \cap \mathcal{U}^{\mathcal{I}_d} = \emptyset$. □

Using Lemma 4, we can perform a further decomposition of each set term appearing in a $\mathcal{ALCCQU}$ role successor constraint into a disjoint union of set terms that have a structure that is similar to qualified number restrictions in $\mathcal{ALCQ}t$.

Lemma 5. *If s is a set term over role names in N_R and $\mathcal{ALCCQU}$ concept descriptions, then s is equivalent to a disjoint union*

$$\bigcup_{i=1}^{N} \omega_i \cap C_i \tag{3.13}$$

where ω_i is a safe role type over N_R and C_i is a $\mathcal{ALCCQU}$ concept description.

Proof. Let $\mathcal{I}$ be a $\mathcal{ALCCQU}$ interpretation and $d \in \Delta^{\mathcal{I}}$. According to $\mathcal{ALCCQU}$ semantics, the inclusion $s^{\mathcal{I}_d} \subseteq \mathcal{U}^{\mathcal{I}_d} = \bigcup_{r \in N_R} r^{\mathcal{I}}(d)$ holds. Hence, we can replace the set term s with the equivalent set term $s' := s \cap \bigcup_{r \in N_R} r$.

By transforming s' into its set-theoretical disjunctive normal form $\mathtt{dnf}(s')$, we obtain an equivalent set term of the form

$$\bigcup_{i=1}^{N'} \bigcap_{j=1}^{m_i} s_j$$

such that $t := \bigcap_{j=1}^{m_i} s_j$ contains at least one positive occurrence of a role name in N_R — this is a consequence of the distributivity law for set unions over set intersections. Since t is a set term over role names and $\mathcal{ALCCQU}$ concept descriptions, we can assume that $t = t_R \cap t_C$ with t_R a set intersection containing only role names (or their complement) and t_C a set intersection over $\mathcal{ALCCQU}$ concept descriptions (or their negation).

If t_R contains both r and r^c for some $r \in N_R$, we replace t with $\bot$: for every interpretation $\mathcal{I}$ and $d \in \Delta^{\mathcal{I}}$, it holds that $t^{\mathcal{I}_d} \subseteq (r \cap r^c)^{\mathcal{I}_d} = \emptyset = \bot^{\mathcal{I}_d}$. Otherwise, t_R corresponds to a

union of disjoint and safe role types $\bigcup_{j=1}^{k_i} \omega_j$ — this is true, because there is at least one positive occurrence of a role name r.

Thanks to Lemma 4, it is possible to replace t_C with a $\mathcal{ALCCQU}$ concept description C_i such that $t_C^{\mathcal{I}_d} = C_i^{\mathcal{I}_d}$. This yields the equivalence

$$t = t_R \cap t_C \equiv \bigcup_{j=1}^{k_i} \omega_j \cap C_i.$$

Let s'' be the set term where each term $\bigcap_{j=1}^{m_i} s_j$ is replaced with an equivalent term $\bigcup_{j=1}^{k_i} \omega_j \cap C_i$. If s'' contains two terms $\omega \cap C$ and $\omega \cap D$ with $C \neq D$, we replace them with the equivalent set term $\omega \cap (C \sqcup D)$. Finally, we obtain that s is equivalent to a disjoint union of the form

$$s'' \equiv \bigcup_{i=1}^{N} (\omega_i \cap C_i). \qquad \square$$

The decomposition described in Lemma 5 is worst-case exponential: if $|N_R| = k$, the disjoint union might contain $2^k - 1$ disjoint set terms in the worst case, one for each safe role type over N_R; moreover, the transformation to the set-theoretic disjunctive normal form is also worst-case exponential.

Example 5. If $N_R = \{r, s\}$, applying the encoding described in Lemma 5 to the set term $A \cup (r^c \cap B)$ yields the disjoint union

$$(r \cap s \cap A) \cup (r \cap s^c \cap A) \cup (r^c \cap s \cap (A \sqcup B)).$$

Theorem 6. *Every role successor constraint in $\mathcal{ALCCQU}$ of the form* $\mathrm{succ}(|s| \geq N)$ *or* $\mathrm{succ}(|s| \leq N)$ *is equivalent to some $\mathcal{ALCQ}t$ concept description.*

Proof. Let $\mathrm{succ}(|s| \geq N)$ be a role successor constraint. As a consequence of Lemma 5 and the fact that the cardinality of a disjoint union of sets is equal to the sum of the cardinalities of each disjunct, it follows that

$$\mathrm{succ}(|s| \geq N) \equiv \mathrm{succ}(\left|\bigcup_{i=1}^{M} (\omega_i \cap C_i)\right| \geq N) \equiv \mathrm{succ}(\sum_{i=1}^{M} |\omega_i \cap C_i| \geq N).$$

Applying the replacement rule (3.7) we obtain that

$$\mathrm{succ}(\sum_{i=1}^{M} |\omega_i \cap C_i| \geq N) \equiv \bigsqcup \{\prod_{i=1}^{M} \mathrm{succ}(|\omega_i \cap C_i| \geq N_i) \mid \sum_{i=1}^{M} N_i = N\}.$$

Every role successor constraint generated by the replacement is of the form $\mathrm{succ}(|\omega \cap C| \geq N)$; thus, we can apply Lemma 3 and obtain the equivalent $\mathcal{ALCQ}t$ concept description

$$\bigsqcup \{\prod_{i=1}^{M'} (\geq N_i\ \omega_i.\ C_i^{\sharp}) \mid \sum_{i=1}^{M'} N_i = N\}$$

where $C_i^\sharp$ denotes the recursive replacement of any $\mathcal{ALCCQU}$ role successor constraint occurring in C_i with its equivalent $\mathcal{ALCQ}t$ concept description. The transformation is guaranteed to terminate, because in each recursive application the nesting level of $\mathcal{ALCCQU}$ role successor constraint is strictly decreasing and the nesting level is bounded by the nesting level of $\mathrm{succ}(|s| \geq N)$, which is finite. We can show that $\mathrm{succ}(|s| \leq N)$ can be encoded in $\mathcal{ALCQ}t$ in a similar way. □

The encoding that we describe in the proof of Theorem 6 is worst-case double exponential in the size of the set term s and the value N appearing in the input role successor constraint. The first exponent is given by the conversion to set-theoretical disjunctive normal form described in Lemma 5, while the second exponent is caused by the application of (3.7) which, as described in Theorem 5, could yield exponentially many new role successor constraints. This encoding is inefficient from a practical point of view and does not constitute a convenient way to use $\mathcal{ALCQ}t$ reasoning services to decide concept satisfiability in $\mathcal{ALCCQU}$ reasoning services to $\mathcal{ALCQ}t$ ones, but it is useful to characterize $\mathcal{ALCCQU}$ as a sublogic of $\mathcal{ALCQ}t$.

Example 6. The $\mathcal{ALCCQU}$ role successor constraint $\mathrm{succ}(|A \cup (r^c \cap B)| \geq 2)$ is equivalent to the $\mathcal{ALCQ}t$ concept description

$$\bigsqcup_{N_1+N_2+N_3=2} ((\geq N_1\ (r \sqcap s).\ A) \sqcap (\geq N_2\ (r \sqcap \neg s).\ A) \sqcap (\geq N_3\ (\neg r \sqcap s).\ A \sqcup B)).$$

Combining the results obtained so far in Lemma 3, Theorem 5 and Theorem 6, we finally obtain the desired corollary.

Corollary 1. *The description logics $\mathcal{ALCCQU}$ and $\mathcal{ALCQ}t$ are equivalent.*

Using this property, we proceed by showing a characterization of $\mathcal{ALCQ}t$ concept descriptions as a specific fragment of first-order logic that can be transferred to $\mathcal{ALCCQU}$, thanks to their equivalence as logics.

3.3 Model-theoretic characterization of $\mathcal{ALCQ}t$

By using the theoretical framework presented in [16], we are able to provide a specific characterization of $\mathcal{ALCQ}t$ concept descriptions. In order to achieve this goal, we first define a notion of $\mathcal{ALCQ}t$-bisimulation that generalizes the counting bisimulation introduced in Section 2.3 by replacing role names with safe role types. We proceed by proving that $\mathcal{ALCQ}t$-bisimilarity implies $\mathcal{ALCQ}t$-equivalence for individuals in different interpretations; for the class of ω-saturated interpretations introduced in Chapter 2 we show that the converse is also true. Lastly, we prove that $\mathcal{ALCQ}t$— thus, $\mathcal{ALCCQU}$— concept descriptions are characterized as the first-order fragment that is invariant under $\mathcal{ALCQ}t$-bisimulation.

The proofs shown in this section are based on the work shown in the extended version of [16].

3.3.1 $\mathcal{ALCQ}t$-bisimulation and invariance for $\mathcal{ALCQ}t$

We define a *bisimulation* for $\mathcal{ALCQ}t$ as a generalization of the *counting bisimulation* introduced in [16] and previously mentioned in Section 2.3.

$\mathcal{ALCQ}t$-BISIMULATION. Given two interpretations $\mathcal{I}_1$ and $\mathcal{I}_2$, the relation $\rho \subseteq \Delta^{\mathcal{I}_1} \times \Delta^{\mathcal{I}_2}$ is a *$\mathcal{ALCQ}t$-bisimulation* between $\mathcal{I}_1$ and $\mathcal{I}_2$ if the following conditions are satisfied:

Atomic $d_1\ \rho\ d_2$ implies

$$d_1 \in A^{\mathcal{I}_1} \text{ if and only if } d_2 \in A^{\mathcal{I}_2}$$

for all $d_1 \in \Delta^{\mathcal{I}_1}$, $d_2 \in \Delta^{\mathcal{I}_2}$ and $A \in N_C$.

Forth if $d_1\ \rho\ d_2$ and $D_1 \subseteq \omega^{\mathcal{I}_1}(d_1)$ is finite for a safe role type ω over N_R, then there is a set $D_2 \subseteq \omega^{\mathcal{I}_2}(d_2)$ such that ρ contains a bijection between D_1 and D_2.

Back if $d_1\ \rho\ d_2$ and $D_2 \subseteq \omega^{\mathcal{I}_2}(d_2)$ is finite for a safe role type ω over N_R, then there is a set $D_1 \subseteq \omega^{\mathcal{I}_1}(d_1)$ such that ρ contains a bijection between D_1 and D_2.

The individuals $d_1 \in \Delta^{\mathcal{I}_1}$ and $d_2 \in \Delta^{\mathcal{I}_2}$ are: *$\mathcal{ALCQ}t$-bisimilar* — notation $(\mathcal{I}_1, d_1) \sim_{\mathcal{ALCQ}t} (\mathcal{I}_2, d_2)$ — if there is a $\mathcal{ALCQ}t$-bisimulation ρ between $\mathcal{I}_1$ and $\mathcal{I}_2$ such that $d_1\ \rho\ d_2$; *$\mathcal{ALCQ}t$-equivalent* — notation $(\mathcal{I}_1, d_1) \equiv_{\mathcal{ALCQ}t} (\mathcal{I}_2, d_2)$ — if for all $\mathcal{ALCQ}t$ concept descriptions C, $d_1 \in C^{\mathcal{I}_1}$ if and only if $d_2 \in C^{\mathcal{I}_2}$.

The first property to prove is that two individuals that are related by a $\mathcal{ALCQ}t$-bisimulation are $\mathcal{ALCQ}t$-equivalent.

Theorem 7. *If* $(\mathcal{I}_1, d_1) \sim_{\mathcal{ALCQ}t} (\mathcal{I}_2, d_2)$ *then* $(\mathcal{I}_1, d_1) \equiv_{\mathcal{ALCQ}t} (\mathcal{I}_2, d_2)$.

Proof. Due to $(\mathcal{I}_1, d_1) \sim_{\mathcal{ALCQ}t} (\mathcal{I}_2, d_2)$, there exists a $\mathcal{ALCQ}t$-bisimulation $\rho \subseteq \Delta^{\mathcal{I}_1} \times \Delta^{\mathcal{I}_2}$ satisfying $d_1\ \rho\ d_2$. We prove that for all $\mathcal{ALCQ}t$ concept descriptions C and $(e_1, e_2) \in \rho$,

$$e_1 \in C^{\mathcal{I}_1} \text{ if and only if } e_2 \in C^{\mathcal{I}_2} \tag{3.14}$$

by structural induction over C.

Concept names If $C = A \in N_C$ then (3.14) follows immediately from $d_1\ \rho\ d_2$, since ρ satisfies the atomic condition for an $\mathcal{ALCQ}t$-bisimulation.

Inductive Hypothesis Assume that, given a $\mathcal{ALCQ}t$ concept description C, (3.14) holds for all proper subconcepts D of C.

Negation Let $C = \neg D$. Then, $d_1 \in C^{\mathcal{I}_1}$ iff $d_1 \in \Delta^{\mathcal{I}_1}$ and $d_1 \notin D^{\mathcal{I}_1}$; since (3.14) holds for D by inductive hypothesis, $d_1 \notin D^{\mathcal{I}_1}$ iff $d_2 \notin D^{\mathcal{I}_2}$. Thus, $d_2 \in \Delta^{\mathcal{I}_2}$ and $d_2 \notin D^{\mathcal{I}_2}$, which is equivalent to $d_2 \in C^{\mathcal{I}_2}$.

Conjunction Let $C = D \sqcap E$. Then, $d_1 \in C^{\mathcal{I}_1}$ iff $d_1 \in D^{\mathcal{I}_1}$ and $d_1 \in E^{\mathcal{I}_1}$. By inductive hypothesis, (3.14) holds for both D and E; thus, $d_1 \in D^{\mathcal{I}_1}$ and $d_1 \in E^{\mathcal{I}_1}$ iff $d_2 \in D^{\mathcal{I}_2}$ and $d_2 \in E^{\mathcal{I}_2}$, hence $d_2 \in C^{\mathcal{I}_2}$. Thanks to the semantic equivalence $D \sqcup E \equiv \neg(\neg D \sqcap \neg E)$ it follows that (3.14) also holds for $C = D \sqcup E$.

Qualified number restriction Assume that $C = (\geq N\ \omega.\ E)$, with $N \geq 0$ and $d_1 \in C^{\mathcal{I}_1}$. Let $E_1 := \omega^{\mathcal{I}_1}(d_1) \cap E^{\mathcal{I}_1}$; then, $|E_1| \geq N$. Let $D_1 \subseteq E_1$ be a subset of cardinality N. Since $D_1 \subseteq \omega^{\mathcal{I}_1}(d_1)$ is finite, $d_1\ \rho\ d_2$ and ρ satisfies the forth condition for a $\mathcal{ALCQ}t$-bisimulation there exists a set $D_2 \subseteq \omega^{\mathcal{I}_2}(d_2)$ in bijection with D_1 under ρ. Since D_2 is the image of D_1 under ρ and every element of D_1 belongs to $E^{\mathcal{I}_1}$, by inductive hypothesis we deduce that $D_2 \in E^{\mathcal{I}_2}$, thus $D_2 \subseteq \omega^{\mathcal{I}_2}(d_2) \cap E^{\mathcal{I}_2}$. We can then conclude that $d_2 \in C^{\mathcal{I}_2}$, since $\left|\omega^{\mathcal{I}_2}(d_2) \cap E^{\mathcal{I}_2}\right| \geq |D_2| = N$. Using the back condition and a similar argument, we can prove that if $d_2 \in C^{\mathcal{I}_2}$, then $d_1 \in C^{\mathcal{I}_1}$. Thanks to the semantic equivalence $(\leq N\ \omega.\ E) \equiv \neg(\geq N+1\ \omega.\ E)$, we obtain that (3.14) holds for $C = (\leq N\ \omega.\ E)$ as well. □

Thanks to Corollary 1, we obtain the following property, where the notion of $\mathcal{ALCCQU}$-equivalence is defined analogously to that of $\mathcal{ALCQ}t$-equivalence.

Corollary 2. *If* $(\mathcal{I}_1, d_1) \sim_{\mathcal{ALCQ}t} (\mathcal{I}_2, d_2)$ *then* $(\mathcal{I}_1, d_1) \equiv_{\mathcal{ALCCQU}} (\mathcal{I}_2, d_2)$.

It is worth mentioning that the property of invariance under $\mathcal{ALCQ}t$-bisimulation might hold for DLs that are more expressive than $\mathcal{ALCQ}t$. Indeed, as we are going to explain in Section 3.5, two individuals that are $\mathcal{ALCQ}t$-bisimilar are also $\mathcal{ALCSCC}^{\infty}$-equivalent; the fact that $\mathcal{ALCSCC}^{\infty}$ is strictly more expressive than $\mathcal{ALCQ}t$ is shown in Section 3.4.

3.3.2 Characterization of $\mathcal{ALCQ}t$ concept descriptions

To begin this section, we show how to embed $\mathcal{ALCQ}t$ in first-order logic by means of a translation mapping. This implies that both the logics $\mathcal{ALCQ}t$ and $\mathcal{ALCCQU}$ are fragments of first-order logic. After that, we prove that for the class of ω-saturated interpretations, $\mathcal{ALCQ}t$-equivalence implies $\mathcal{ALCQ}t$-bisimilarity. In this way, we obtain a different but equivalent way to distinguish individuals in different interpretations, according to the set of $\mathcal{ALCQ}t$ concepts that they satisfy.

At the end of this section, we show that the DLs $\mathcal{ALCQ}t$ and $\mathcal{ALCCQU}$ can be characterized as the first-order fragment that is invariant under $\mathcal{ALCQ}t$-bisimulation, providing a stronger definition for $\mathcal{ALCCQU}$ than a simple syntactic restriction of $\mathcal{ALCSCC}^{\infty}$.

First-order translation of $\mathcal{ALCQ}t$. We define a first-order role type translation $\pi_{x,y}$ that maps role types to first-order formulas with free variables x and y, where $r \in N_R$ and ω_1, ω_2 are safe role types:

$$\pi_{x,y}(r) := r(x,y), \quad \pi_{x,y}(\neg r) := \neg r(x,y), \quad \pi_{x,y}(\omega_1 \sqcap \omega_2) := \pi_{x,y}(\omega_1) \wedge \pi_{x,y}(\omega_2).$$

We then proceed to define a first-order concept translation π_x that maps $\mathcal{ALCQ}t$ concept descriptions to first-order formulas with free variable x:

$$\begin{aligned}
\pi_x(A) &:= A(x) \quad A \in N_C & \pi_x(\neg C) &:= \neg \pi_x(C) \\
\pi_x(C \sqcap D) &:= \pi_x(C) \wedge \pi_x(D) & \pi_x(C \sqcup D) &:= \pi_x(C) \vee \pi_x(D) \\
\pi_x((\geq N\ \omega.\ C)) &:= \exists x_1. \cdots \exists x_N. \bigwedge_{i=1}^{N} \bigwedge_{j=i+1}^{N} x_i \neq x_j \wedge \bigwedge_{i=1}^{N} (\pi_{x,x_i}(\omega) \wedge \pi_{x_i}(C))
\end{aligned}$$

We do not explicitly introduce a first-order translation for $(\leq N\ \omega.\ C)$, since $(\leq N\ \omega.\ C) \equiv \neg(\geq N+1\ \omega.\ C)$ holds.

Lemma 6. *For every $\mathcal{ALCQ}t$ role type ω and interpretation $\mathcal{I}$*

$$(d,e) \in \omega^{\mathcal{I}} \textit{ if and only if } \mathcal{I}, \{x/d, y/e\} \models \pi_{x,y}(\omega).$$

Proof. Holds trivially. □

Theorem 8. *For every $\mathcal{ALCQ}t$ concept description C and interpretation $\mathcal{I}$*

$$d \in C^{\mathcal{I}} \textit{ if and only if } \mathcal{I}, \{x/d\} \models \pi_x(C).$$

Proof. By structural induction over the $\mathcal{ALCQ}t$ concept description C.

Concept names Assume that $C = A \in N_C$. Then, $d \in A^{\mathcal{I}}$ iff $\mathcal{I}, \{x/d\} \models A(x)$ follows from the definition of $\mathcal{I}$ as a first-order interpretation.

Inductive Hypothesis Assume that the claim holds for all the proper subconcepts of C.

Negation Assume that $C = \neg D$. Then, $d \in C^{\mathcal{I}}$ iff $d \notin D^{\mathcal{I}}$ iff $\mathcal{I}, \{x/d\} \not\models \pi_x(D)$ iff $\mathcal{I}, \{x/d\} \models \pi_x(C)$.

Conjunction Assume that $C = D \sqcap E$. Then, $d \in C^{\mathcal{I}}$ iff $d \in D^{\mathcal{I}}$ and $d \in E^{\mathcal{I}}$ iff $\mathcal{I}, \{x/d\} \models \pi_x(D)$ and $\mathcal{I}, \{x/d\} \models \pi_x(E)$ iff $\mathcal{I}, \{x/d\} \models \pi_x(D) \wedge \pi_x(E)$. Since $D \sqcup E \equiv \neg(\neg D \sqcap \neg E)$ and $\varphi \vee \psi \equiv \neg(\neg\varphi \wedge \neg\psi)$, this is sufficient to show that the claim holds for the case of disjunction.

Number restrictions Assume that $d \in (\geq N\ \omega.\ C)^{\mathcal{I}}$. Then, there exist N distinct individuals $d_1, \ldots, d_N \in \Delta^{\mathcal{I}}$ such that $(d, d_i) \in \omega^{\mathcal{I}}$ and $d_i \in C^{\mathcal{I}}$ for $i = 1, \ldots, N$. Thanks to Lemma 6 and the inductive hypothesis, it follows that $\mathcal{I}, \{x/d, x_i/d_i\} \models \pi_{x,x_i}(\omega)$ and $\mathcal{I}, \{x_i/d_i\} \models C^{\mathcal{I}}$ for $i = 1, \ldots, N$. Moreover, if $d_i \neq d_j$, the formula $x_i \neq x_j$ is satisfied by the assignment $\{x_i/d_i, x_j/d_j\}$. Therefore, $\mathcal{I}, \{x/d\} \models \pi_x((\geq N\ \omega.\ C))$. We can show the opposite implication in a similar way. Since $(\leq N\ \omega.\ C) \equiv \neg(\geq N+1\ \omega.\ C)$, this is sufficient to prove that the claim holds for all possible number restrictions. □

N-ary existential quantification in $\mathcal{ALCQ}t$. It is possible to add to $\mathcal{ALCQ}t$ an additional construct similar to the N-ary existential quantification introduced in [5], without increasing the expressive power. The intuitive semantics of the construct $\exists\omega.(C_1, \ldots, C_N)$ is that there are N distinct ω-successors $d_1, \ldots, d_N$ and that d_i is described by C_i for $i = 1, \ldots, N$. We use this construct to prove that, if we restrict our attention to ω-saturated interpretations from Definition 1, $\mathcal{ALCQ}t$-equivalent individuals are $\mathcal{ALCQ}t$-bisimilar.

Let ω be a safe role over N_R, $N \geq 1$ a natural number and $C_1, \ldots, C_N$ $\mathcal{ALCQ}t$ concept description. Then, the semantics of the N-ary existential quantification $\exists\omega.(C_1, \ldots, C_N)$ under an interpretation $\mathcal{I}$ is defined as

$$\exists\omega.(C_1, \ldots, C_N)^{\mathcal{I}} := \Big\{ x \in \Delta^{\mathcal{I}} \mid \exists y_1, \ldots, y_n \in \Delta^{\mathcal{I}}.(x, y_i) \in \omega^{\mathcal{I}} \wedge \\ \wedge\, y_i \in C_i^{\mathcal{I}} \text{ for } i = 1, \ldots, N \wedge \bigwedge_{1 \leq i < j \leq N} y_i \neq y_j \Big\}. \tag{3.15}$$

The first-order translation of the N-ary existential quantification can be defined as:

$$\pi_x(\exists\omega.(C_1, \ldots, C_N)) := \exists x_1, \ldots, x_N.(\bigwedge_{i=1}^{N}(\pi_{x,x_i}(\omega) \wedge \pi_{x_i}(C_i)) \wedge \bigwedge_{1 \leq i < j \leq N} x_i \neq x_j). \tag{3.16}$$

The proof that the N-ary existential quantification can be expressed as a proper $\mathcal{ALCQ}t$ concept description relies on the notion of a *system of distinct representatives* for the sets $C_1, \ldots, C_N$ — distinct individuals $d_1, \ldots, d_N$ such that $d_i \in C_i$ for $i = 1, \ldots, N$ — and the following characterization by Hall [13].

Theorem 9 (Hall)**.** *The sets $C_1, \ldots, C_N$ possess a system of distinct representatives if and only if $\left|\bigcup_{j=1}^{k} C_{i_j}\right| \geq k$ for each subset $\{i_1, \ldots, i_k\} \subseteq \{1, \ldots, N\}$ of distinct indexes $i_1, \ldots, i_k$.*

By using Theorem 9 and adapting the proof shown in [5] to use safe role types instead of role names, we obtain the following result.

Theorem 10. *The N-ary existential quantification constructor can be expressed in $\mathcal{ALCQ}t$, in particular*

$$\exists\omega.(C_1, \ldots, C_N)^{\mathcal{I}} \equiv \bigsqcap\{(\geq k\ \omega.\ C_{i_1} \sqcup \cdots \sqcup C_{i_k}) \mid \{i_1, \ldots, i_k\} \subseteq \{1, \ldots, N\}\}. \tag{3.17}$$

Proof. Omitted. For details, refer to [5]. □

Using Theorem 10, we show that two $\mathcal{ALCQ}t$-equivalent individuals are also $\mathcal{ALCQ}t$-bisimilar. In the following proof, for every set of first-order formulae Γ we denote with $\Gamma[e_2/e_1]$ the set obtained by replacing every occurrence of the constant e_2 with e_1 in Γ.

Theorem 11. *For all ω-saturated interpretations $\mathcal{I}_1$, $\mathcal{I}_2$,*

$$\textit{if } (\mathcal{I}_1, d_1) \equiv_{\mathcal{ALCQ}t} (\mathcal{I}_2, d_2) \textit{ and } \mathcal{I}_1, \mathcal{I}_2 \textit{ then } (\mathcal{I}_1, d_1) \sim_{\mathcal{ALCQ}t} (\mathcal{I}_2, d_2).$$

Proof. We show that the relation

$$S := \{(e_1, e_2) \in \Delta^{\mathcal{I}_1} \times \Delta^{\mathcal{I}_2} \mid e_1 \equiv_{\mathcal{ALCQ}t} e_2\}$$

is a $\mathcal{ALCQ}t$-bisimulation such that $(d_1, d_2) \in S$. The atomic condition is trivially satisfied by S, by definition. To show that the forth condition of $\mathcal{ALCQ}t$-bisimulation is satisfied by S, assume

that $(e_1, e_2) \in S$ and let $D_1 = \{d_1, \ldots, d_N\} \subseteq \omega^{\mathcal{I}_1}(d_1)$ be finite, with ω a safe role type over N_R. We introduce a variable x_i for every individual $d_i \in D_1$ and consider the set of first-order formulas $\Gamma := \Gamma^{\neq} \cup \Gamma^{\omega} \cup \bigcup_{i=1}^{N} T_i$, where

$$\begin{aligned}
\Gamma^{\neq} &:= \{\neg(x_i, x_j) \mid d_i, d_j \in D_1 \wedge i \neq j\} && \text{(variables are all distinct)} \\
\Gamma^{\omega} &:= \{\omega(e_2, x_i) \mid d_i \in D_1\} && (\omega\text{-successors of } e_2) \\
T_i &:= \{\pi_{x_i}(C) \mid C \text{ is a } \mathcal{ALCQ}t \text{ concept and } d_i \in C^{\mathcal{I}_1}\} && (\mathcal{ALCQ}t\text{-type of } d)
\end{aligned}$$

The set $\Gamma_1 := \Gamma[e_2/e_1]$ is realizable in $\mathcal{I}_1$ under the variable assignment $\{x_1/d_1, \ldots, x_N/d_N\}$.

We now prove that Γ is realizable in $\mathcal{I}_2$ by using the fact that $\mathcal{I}_2$ is ω-saturated. Let $\Gamma' \subseteq \Gamma$ be a finite subset of Γ. Let $\Gamma'_1 := \Gamma'[e_2/e_1]$; then, Γ'_1 is satisfiable in $\mathcal{I}_1$ under the variable assignment $\{x_1/d_1, \ldots, x_N/d_N\}$, because $\Gamma'_1 \subseteq \Gamma_1$. Without loss of generality, we can assume that Γ'_1 contains $\Gamma_1^{\neq}$ and Γ_1^{ω}, since they are both finite. For every variable x_i with $i = 1, \ldots, N$ let

$$t_i := \{C \mid C \text{ is an } \mathcal{ALCQ}t \text{ concept and } \pi_{x_i}(C) \in \Gamma'_1\}.$$

By defining $C_i := \bigsqcap\{C \mid C \in t_i\}$ and noticing that both $\Gamma^{\neq}$ and Γ_1^{ω} are in Γ'_1, we obtain that

$$\mathcal{I}_1, \{x_1/d_1, \ldots, x_N/d_N\} \models \Gamma^{\neq} \wedge \Gamma_1^{\omega} \wedge \bigwedge_{i=1}^{N} \pi_{x_i}(C_i)$$

hence $\mathcal{I}_1, \{x/e_1\} \models \pi_x(\exists\omega.(C_1, \ldots, C_N))$.

Thanks to Theorem 8 it follows that $e_1 \in \exists\omega.(C_1, \ldots, C_N)^{\mathcal{I}_1}$. We assumed that $e_1 \equiv_{\mathcal{ALCQ}t} e_2$, thus $e_2 \in \exists\omega.(C_1, \ldots, C_N)^{\mathcal{I}_2}$. Using Theorem 8 again, this yields that Γ' is satisfiable in $\mathcal{I}_2$. The interpretation $\mathcal{I}_2$ is ω-saturated by assumption, hence we conclude that Γ is realizable in $\mathcal{I}_2$.

Let $\mathcal{Z}_2$ be a variable assignment such that $\mathcal{I}_2, \mathcal{Z}_2 \models \Gamma$ and let $D_2 := \{\mathcal{Z}_2(x_i) \mid d_i \in D_1\}$. From $\mathcal{I}_2, \mathcal{Z}_2 \models T_i$ it follows that $d_i \equiv_{\mathcal{ALCQ}t} \mathcal{Z}_2(x_i)$; moreover, $D_2 \subseteq \omega^{\mathcal{I}_2}(d_2)$ since $\mathcal{I}_2, \mathcal{Z}_2 \models \Gamma^{\omega}$; finally, the mapping $d_i \mapsto \mathcal{Z}_2(x_i)$ is bijection from D_1 to D_2, thanks to $\Gamma^{\neq}$. Thus, the forth condition holds for S. Using a similar argument, we can show that the back condition holds for S. This allows us to conclude that S is a $\mathcal{ALCQ}t$-bisimulation between $\mathcal{I}_1$ and $\mathcal{I}_2$ such that $(d_1, d_2) \in S$. □

In order to characterize $\mathcal{ALCQ}t$ as the first-order fragment that is invariant under $\mathcal{ALCQ}t$-bisimulation, we make use of Theorem 2, taken from [8], that allows us to consider ω-saturated models for a set of first-order sentences, without loss of generality.

Theorem 12. *Let $\varphi(x)$ be a first-order formula. The following are equivalent:*

1. *there exists a $\mathcal{ALCQ}t$ concept description C such that $\pi_x(C) \equiv \varphi(x)$;*
2. *$\varphi(x)$ is invariant under $\sim_{\mathcal{ALCQ}t}$.*

Proof. Assume that there exists a $\mathcal{ALCQ}t$ concept description C such that $\pi_x(C) \equiv \varphi(x)$. Then, $d_i \in \pi_x(C)^{\mathcal{I}_i}$ iff $d_i \in \varphi^{\mathcal{I}_i}$ for $i \in \{1, 2\}$. Using Theorem 8 we obtain that $d_i \in \pi_x(C)^{\mathcal{I}_i}$ iff $d_i \in C^{\mathcal{I}_i}$. Thanks to Theorem 7, we also know that obtain that if $(\mathcal{I}_1, d_1) \sim_{\mathcal{ALCQ}t} (\mathcal{I}_2, d_2)$, then

$d_1 \in C^{\mathcal{I}_1}$ iff $d_2 \in C^{\mathcal{I}_2}$. Combining all these equivalences, we obtain that if $(\mathcal{I}_1, d_1) \sim_{\mathcal{ALCQ}t} (\mathcal{I}_2, d_2)$, then $d_1 \in \varphi^{\mathcal{I}_1}$ iff $d_2 \in \varphi^{\mathcal{I}_2}$, hence the invariance of $\varphi(x)$ under $\sim_{\mathcal{ALCQ}t}$.

Assume now that $\varphi(x)$ is invariant under $\sim_{\mathcal{ALCQ}t}$ but that there is no $\mathcal{ALCQ}t$ concept description C such that $\pi_x(C) \equiv \varphi(x)$. This implies that $\varphi(x)$ is satisfiable — since $\varphi(x) \not\equiv \bot$ — and that $\neg\varphi(x)$ is also satisfiable — because $\varphi(x) \not\equiv \top$. We define the set of first-order translations of $\mathcal{ALCQ}t$ concepts that are entailed by $\varphi(x)$:

$$\Psi(\varphi(x)) := \{\pi_x(C) \mid C \text{ is a } \mathcal{ALCQ}t \text{ concept}, \varphi(x) \models \pi_x(C)\}$$

and we show that the set of formulae $\Psi(\varphi(x)) \cup \{\neg\varphi(x)\}$ is satisfiable, using the compactness of first-order logic. We notice that for each formula $\psi(x) \in \Psi(\varphi(x))$ there exist an interpretation $\mathcal{I}_\psi$ and $d \in \Delta^{\mathcal{I}_\psi}$ such that $\mathcal{I}_\psi, \{x/d\} \models \psi(x)$ and $\mathcal{I}_\psi, \{x/d\} \models \neg\varphi(x)$, because of our assumption that $\pi_x(C) \not\equiv \varphi(x)$ for all $\mathcal{ALCQ}t$ concepts C.

Every finite subset S of $\Psi(\varphi(x))$ is satisfiable, since $\varphi(x)$ is satisfiable and $\varphi(x) \models \Psi(\varphi(x))$. Moreover, we can assume that there are $\mathcal{I}_S$ and $d \in \Delta^{\mathcal{I}_S}$ such that $\mathcal{I}_S, \{x/d\} \models S$ and $\mathcal{I}_S, \{x/d\} \models \neg\varphi(x)$, otherwise $\varphi(x) \equiv \pi_x(\bigsqcap_{C\in S} C)$ would invalidate our assumption. Hence, every finite subset of $\Psi(\varphi(x)) \cup \{\neg\varphi(x)\}$ is satisfiable. We can conclude, using the compactness of first-order logic, that the set $\Psi(\varphi(x)) \cup \{\neg\varphi(x)\}$ is satisfiable.

Let $\mathcal{I}_2$ be an interpretation satisfying $\mathcal{I}_2, \{x/d_2\} \models \Psi(\varphi(x)) \cup \{\neg\varphi(x)\}$; since $\Psi(\varphi(d_2)) \cup \{\neg\varphi(d_2)\}$ is a first-order sentence, due to Theorem 2 we can assume that $\mathcal{I}_2$ is ω-saturated.

Let $T' := \{\pi_x(C) \mid C \text{ is a } \mathcal{ALCQ}t \text{ concept}, d_2 \in C^{\mathcal{I}_2}\}$. We prove by contradiction that the set of formulae $T := \{\varphi(x)\} \cup T'$ is satisfiable and we prove it by contradiction. If we assume that T is unsatisfiable, as a consequence of the compactness of first-order logic there is a finite set of $\mathcal{ALCQ}t$ concepts Γ such that:

1. $d_2 \in C^{\mathcal{I}_2}$ for all $C \in \Gamma$,
2. $\{\varphi(x)\} \cup \{\pi_x(C) \mid C \in \Gamma\} \subseteq T$ is unsatisfiable.

If we define $D := \bigsqcap_{C\in\Gamma} C$, from the first point follows that $d_2 \in D^{\mathcal{I}_2}$. From the second point and the equality $\bigwedge_{C\in\Gamma} \pi_x(C) = \pi_x(\bigsqcap_{C\in\Gamma} C)$, we obtain the tautology

$$\models \varphi(x) \to \pi_x\big(\neg\bigsqcap\nolimits_{C\in\Gamma} C\big).$$

Using the deduction theorem we reach the conclusion that $\varphi(x) \models \pi_x(\neg D)$, hence $\pi_x(\neg D) \in \Psi(\varphi(x))$. Since $\mathcal{I}_2, \{x/d_2\} \models \Psi(\varphi(x))$ by definition, it follows that for every $\mathcal{ALCQ}t$ concept C such that $\pi_x(C) \in \Psi(\varphi(x))$, $d_2 \in C^{\mathcal{I}_2}$ holds. Hence, $\Psi(\varphi(x)) \subseteq T'$. Since $\pi_x(\neg D) \in \Psi(\varphi(x))$, we deduce that $d_2 \in (\neg D)^{\mathcal{I}_2}$; this, together with the information that $d_2 \in D^{\mathcal{I}_2}$, we reach a contradiction. Therefore, T cannot be unsatisfiable.

Let $\mathcal{I}_1$ be a ω-saturated interpretation satisfying $\mathcal{I}_1, \{x/d_1\} \models T$. From the definition of T follows that $(\mathcal{I}_1, d_1) \equiv_{\mathcal{ALCQ}} t(\mathcal{I}_2, d_2)$; by Theorem 11, this implies that $(\mathcal{I}_1, d_1) \sim_{\mathcal{ALCQ}t} (\mathcal{I}_2, d_2)$. Since $d_1 \in \varphi^{\mathcal{I}_1}$ but $d_2 \notin \varphi^{\mathcal{I}_2}$, we reach a contradiction, since $\varphi(x)$ is invariant under $\mathcal{ALCQ}t$-bisimulation by hypothesis. Therefore, there exists a $\mathcal{ALCQ}t$ concept C such that $\pi_x(C) \equiv \varphi(x)$. □

We thus proved that $\mathcal{ALCQ}t$ and $\mathcal{ALCCQU}$ are characterized as the first-order fragment that is invariant under $\mathcal{ALCQ}t$-bisimulation. This property is going to be used in the next section to

show that there are $\mathcal{ALCSCC}^{\infty}$ concept descriptions that cannot be expressed as first-order formulae. Before that, we classify the DLs $\mathcal{ALCQ}$, $\mathcal{ALCCQU}$, $\mathcal{ALCQ}t$ and $\mathcal{ALCSCC}^{\infty}$ according to their expressive power.

3.4 Expressive power

Thanks to the notions of counting bisimulation and $\mathcal{ALCQ}t$-bisimulation, introduced respectively in Section 2.3 and Section 3.3, we can now classify the DLs analyszed so far according to their expressive power.

3.4.1 Relative expressivity of $\mathcal{ALCQ}$ and $\mathcal{ALCCQU}$

$\mathcal{ALCQ}$ and $\mathcal{ALCCQU}$ with only one role name are equivalent. When N_R contains a unique role name r, the description logic $\mathcal{ALCQ}t$ degenerates to $\mathcal{ALCQ}$. In this case, the only safe role type over $\{r\}$ is r itself and the only $\mathcal{ALCQ}t$ qualified number restrictions that can be expressed are $(\geq N\ r.\ C)$ or $(\leq N\ r.\ C)$, which are also $\mathcal{ALCQ}$ qualified number restrictions. Therefore, thanks to Corollary 1, $\mathcal{ALCQ}$ and $\mathcal{ALCCQU}$ are equivalent when N_R contains only one role name.

$\mathcal{ALCQ}$ is a sublogic of $\mathcal{ALCCQU}$. In the general setting, the description logic $\mathcal{ALCQ}$ is a sublogic of $\mathcal{ALCCQU}$. To prove that this claim holds, we show that every qualified number restriction $(\leq N\ r.\ C)$ and $(\geq N\ r.\ C)$ can be expressed in $\mathcal{ALCCQU}$. In the following lemma, let $C^{\sharp}$ denote the $\mathcal{ALCCQU}$ translation of the $\mathcal{ALCQ}$ concept description C.

Lemma 7. *For every interpretation $\mathcal{I}$ we have*

$$(\geq N\ r.\ C)^{\mathcal{I}} = \mathrm{succ}(\left|r \cap C^{\sharp}\right| \geq N)^{\mathcal{I}} \text{ and } (\leq N\ r.\ C)^{\mathcal{I}} = \mathrm{succ}(\left|r \cap C^{\sharp}\right| \leq N)^{\mathcal{I}}.$$

Proof. Analogous to Lemma 3. □

$\mathcal{ALCCQU}$ is more expressive than $\mathcal{ALCQ}$. As a consequence of Theorem 3, we can show that there are $\mathcal{ALCCQU}$ concept descriptions that are not expressible in $\mathcal{ALCQ}$ by using invariance of $\mathcal{ALCQ}$ concept descriptions under counting bisimulation. We provide appropriate counter-examples, highlighting features of $\mathcal{ALCCQU}$ that are not expressible in $\mathcal{ALCQ}$.

Corollary 3. *$\mathcal{ALCQ}$ cannot express* local role disjointness*, that is, there is no $\mathcal{ALCQ}$-concept C such that $C \equiv \mathrm{succ}(|r \cap s| = 0)$.*

Proof. If a $\mathcal{ALCQ}$ concept description C such that $C \equiv \mathrm{succ}(|r \cap s| = 0)$ existed, we would obtain a contradiction. To see this, we consider the interpretations $\mathcal{I}_0$ and $\mathcal{I}_1$ shown in Figure 3.1. The relation

$$\rho := \{(d_0, d_1), (e_0, e_1), (e_0, f_1)\}$$

satisfies the axioms of Definition 2, hence $(\mathcal{I}_0, d_0) \sim_{\mathcal{ALCQ}} (\mathcal{I}_1, d_1)$. Since $d_1 \in \mathrm{succ}(|r \cap s| = 0)^{\mathcal{I}_1}$, it follows that $d_1 \in C^{\mathcal{I}_1}$; thanks to Theorem 3, this implies that $d_0 \in C^{\mathcal{I}_0}$. However, $d_0 \notin \mathrm{succ}(|r \cap s| = 0)^{\mathcal{I}_0}$; hence, we would obtain a contradiction. □

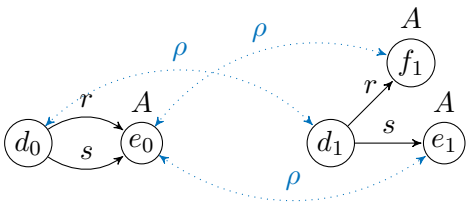

Figure 3.1: Two interpretations $\mathcal{I}_0$ and $\mathcal{I}_1$, depicted as graphs, that are $\mathcal{ALCQ}$-bisimilar under ρ — represented by means of dotted arrows.

Corollary 4. *$\mathcal{ALCQ}$ cannot express the fact that all the successors of a certain individual belonging to a concept name A must also be filler for a role name r, that is, there is no $\mathcal{ALCQ}$ concept C such that $C \equiv \mathrm{succ}(A \subseteq r)$.*

Proof. If a $\mathcal{ALCQ}$-concept description C such that $C \equiv \mathrm{succ}(A \subseteq r)$ existed, we would obtain a contradiction. To see this, we consider the interpretations $\mathcal{I}_0$ and $\mathcal{I}_1$ shown in Figure 3.1. The relation

$$\rho := \{(d_0, d_1), (e_0, e_1), (e_0, f_1)\}$$

is an $\mathcal{ALCQ}$ bisimulation, thus $(\mathcal{I}_0, d_0) \sim_{\mathcal{ALCQ}} (\mathcal{I}_1, d_1)$ and $d_0 \in C^{\mathcal{I}_0}$ iff $d_1 \in C^{\mathcal{I}_1}$. This would contradict our hypothesis, since $d_0 \in \mathrm{succ}(A \subseteq r)^{\mathcal{I}_0}$ but $d_1 \notin \mathrm{succ}(A \subseteq r)^{\mathcal{I}_1}$. □

3.4.2 Relative expressivity of $\mathcal{ALCCQU}$ and $\mathcal{ALCSCC}^\infty$

$\mathcal{ALCCQU}$ is a sublogic of $\mathcal{ALCSCC}^\infty$. The fact that $\mathcal{ALCCQU}$ is a sublogic of $\mathcal{ALCSCC}^\infty$ is a consequence of the definition of the two DLs. Indeed, $\mathcal{ALCCQU}$ corresponds to the subset of $\mathcal{ALCSCC}^\infty$ where every role successor constraint can only contain a cardinality constraint that compare a complex cardinality term with a natural number. Since in our setting we only allow for addition of cardinality terms and we disallow for other arithmetic operations, it is impossible to reduce a cardinality constraint of the form $k = \ell$ with both k and ℓ complex cardinality constraints to a comparison that is acceptable in $\mathcal{ALCCQU}$.

$\mathcal{ALCSCC}^\infty$ is more expressive than $\mathcal{ALCCQU}$. To prove that $\mathcal{ALCSCC}^\infty$ contains concept descriptions that cannot be expressed in $\mathcal{ALCCQU}$, we first show that $\mathcal{ALCSCC}^\infty$ is more expressive than $\mathcal{ALCQ}t$ and then transfer the result to $\mathcal{ALCCQU}$ using Corollary 1. The proof is an extension of the one used in [2] to show that $\mathcal{ALCSCC}^\infty$ is more expressive than $\mathcal{ALCQ}$.

Theorem 13. *The $\mathcal{ALCSCC}^\infty$ concept description $C := \mathrm{succ}(|r \cap A| = |r \cap \neg A|)$ with $A \in N_C$ cannot be expressed in $\mathcal{ALCQ}t$. There exists a $\mathcal{ALCSCC}^\infty$ concept description C that cannot be expressed in $\mathcal{ALCQ}t$, that is, for every $\mathcal{ALCQ}t$ concept D there exists an interpretation $\mathcal{I}$ such that $C^{\mathcal{I}} \neq D^{\mathcal{I}}$.*

Proof. We show by contradiction that for every $\mathcal{ALCQ}t$ concept description D there is an interpretation $\mathcal{J}$ such that $C^{\mathcal{J}} \neq D^{\mathcal{J}}$. Assume that there exists a $\mathcal{ALCQ}t$ concept D_0 such that $C \equiv_{\mathcal{ALCQ}t} D_0$; w.l.o.g. we can assume that every qualified number restriction appearing in D_0 is of the form $(\geq K\ \omega.\ E)$ with ω a safe role type, since $(\leq K\ \omega.\ E) \equiv \neg(\geq K + 1\ \omega.\ E)$. Let

N' be the largest natural number appearing in a role successor constraint in D_0 and $N := N'+1$. We define the interpretation $\mathcal{I}$ with domain $\Delta^\mathcal{I} = \mathbb{N}$ where the only concept or role names interpreted as non-empty sets are

$$r^\mathcal{I} = \{(0, i) \mid i = 1, \ldots, N\}, \quad A^\mathcal{I} = \{1, \ldots, N\}.$$

Then, $0 \in C^\mathcal{I}$ and by assumption of equivalence $0 \in D^\mathcal{I}$.

We observe that for $1 \leq i, j \leq N$, $(\mathcal{I}, i) \sim_{\mathcal{ALCQ}t} (\mathcal{I}, j)$: this is true because none of these individuals has role successors — thus the back and forth conditions of $\mathcal{ALCQ}t$-bisimulation are vacuously true — and the atomic condition is trivially satisfied, since $i, j \in A^\mathcal{I}$ and $i, j \notin B^\mathcal{I}$ for each concept name $B \neq A$. In a similar way, we can argue that $(\mathcal{I}, N + i) \sim_{\mathcal{ALCQ}t} (\mathcal{I}, N + j)$ for $1 \leq i, j \leq N$. Thus, for every concept description E, we obtain the following cases:

1. $\{1, \ldots, N\} \subseteq E^\mathcal{I}$ and $\{N + 1, \ldots, 2N\} \subseteq E^\mathcal{I}$
2. $\{1, \ldots, N\} \subseteq E^\mathcal{I}$ and $\{N + 1, \ldots, 2N\} \subseteq (\neg E)^\mathcal{I}$
3. $\{1, \ldots, N\} \subseteq (\neg E)^\mathcal{I}$ and $\{N + 1, \ldots, 2N\} \subseteq E^\mathcal{I}$
4. $\{1, \ldots, N\} \subseteq (\neg E)^\mathcal{I}$ and $\{N + 1, \ldots, 2N\} \subseteq (\neg E)^\mathcal{I}$

Let $\mathcal{I}'$ be the interpretation obtained by extending $\mathcal{I}$ as follows:

$$r^{\mathcal{I}'} = r^\mathcal{I} \cup \{(0, 2N + 1)\}, \quad A^{\mathcal{I}'} := A^\mathcal{I} \cup \{2N + 1\}.$$

We observe that in $\mathcal{I}'$, $(\mathcal{I}', i) \sim_{\mathcal{ALCQ}t} (\mathcal{I}', j)$ and $(\mathcal{I}', i) \sim_{\mathcal{ALCQ}t} (\mathcal{I}, 2N+1)$ for $1 \leq i, j \leq N$.

Let ω_r be the safe role type where only the role name r occurs positively. For $\omega \neq \omega_r$, we have that $(\geq K\ \omega.\ E)^\mathcal{I} = (\geq K\ \omega.\ E)^{\mathcal{I}'}$ because we only add r-successors. We show that $0 \in (\geq K\ \omega_r.\ E)^\mathcal{I}$ if and only if $0 \in (\geq K\ \omega_r.\ E)^{\mathcal{I}'}$ by case analysis. In case 1 and 2, we obtain that $0 \in (\geq K\ \omega_r.\ E)^\mathcal{I}$ because $\{1, \ldots, N\} \subseteq E^\mathcal{I}$; since $|\{1, \ldots, N\} \cup \{2N + 1\}| = N+1 \geq K$ and $2N+1 \in E^\mathcal{I}$ (due to $\mathcal{ALCQ}t$-bisimilarity) it follows that $0 \in (\geq K\ \omega_r.\ E)^{\mathcal{I}'}$. In case 3, we obtain that $0 \in (\geq K\ \omega_r.\ E)^\mathcal{I}$ because $\{N + 1, \ldots, 2N\} \subseteq E^\mathcal{I}$; since $\{N + 1, \ldots, 2N\} \subseteq E^{\mathcal{I}'}$, it follows that $0 \in (\geq K\ \omega_r.\ E)^{\mathcal{I}'}$. To show that in case 4 our claim holds, we distinguish two additional cases for K. If $K = 0$, then $(\geq 0\ \omega_r.\ E) \equiv \top$, hence the claim follows trivially. If $K \geq 1$, then $0 \notin (\geq K\ \omega_r.\ E)^\mathcal{I}$ since $r^\mathcal{I}(0) \cap E^\mathcal{I} = \emptyset$. Thanks to $\mathcal{ALCQ}t$-bisimilarity, it holds that $2N + 1 \notin E^\mathcal{I}$, therefore $0 \notin (\geq K\ \omega_r.\ E)^{\mathcal{I}'}$. We thus proved that $0 \in (\geq K\ \omega_r.\ E)^\mathcal{I}$ if and only if $0 \in (\geq K\ \omega_r.\ E)^{\mathcal{I}'}$. It follows that $0 \in D_0^{\mathcal{I}'}$. However, we reach a contradiction: since $|r \cap A|^{\mathcal{I}'} = N + 1$ and $|r \cap \neg A|^{\mathcal{I}'} = N$, it follows that $0 \notin C^{\mathcal{I}'}$; but we assumed that $C \equiv D_0$, hence $0 \in C^{\mathcal{I}'}$ also holds. We therefore conclude that C cannot be equivalent to a $\mathcal{ALCQ}t$ concept description D_0. □

3.5 $\mathcal{ALCCQU}$ AS THE FIRST-ORDER FRAGMENT OF $\mathcal{ALCSCC}^\infty$

Despite having a greater expressive power than $\mathcal{ALCCQU}$, $\mathcal{ALCSCC}^\infty$ turns out to be indistinguishable from $\mathcal{ALCCQU}$ under $\mathcal{ALCQ}t$-bisimulation: indeed, in this section we show that all

$\mathcal{ALCSCC}^\infty$ concepts are invariant under $\mathcal{ALCQ}t$-bisimulation. This explains why we need to resort to a different argument in order to prove that $\mathcal{ALCSCC}^\infty$ is more expressive than $\mathcal{ALCCQU}$. Using the characterization result provided by Theorem 12, we deduce that $\mathcal{ALCSCC}^\infty$ contains concept descriptions that are inexpressible in first-order logic.

To prove that all $\mathcal{ALCSCC}^\infty$ concept descriptions are invariant under $\mathcal{ALCQ}t$-bisimulation, we use the decomposition of set terms detailed in Lemma 5.

Theorem 14. *If* $(\mathcal{I}_1, d_1) \sim_{\mathcal{ALCQ}t} (\mathcal{I}_2, d_2)$ *then* $(\mathcal{I}_1, d_1) \equiv_{\mathcal{ALCSCC}} (\mathcal{I}_2, d_2)$.

Proof. Due to $(\mathcal{I}_1, d_1) \sim_{\mathcal{ALCQ}t} (\mathcal{I}_2, d_2)$, there exists a $\mathcal{ALCQ}t$-bisimulation $\rho \subseteq \Delta^{\mathcal{I}_1} \times \Delta^{\mathcal{I}_2}$ satisfying $d_1 \ \rho \ d_2$. We prove that, given two complex cardinality terms k and ℓ over role names and $\mathcal{ALCSCC}$ concept descriptions and $(e_1, e_2) \in \rho$,

$$e_1 \in \mathrm{succ}(k \bowtie \ell)^{\mathcal{I}_1} \text{ if and only if } e_2 \in \mathrm{succ}(k \bowtie \ell)^{\mathcal{I}_2} \tag{3.18}$$

with $\bowtie \in \{\leq, =, \geq\}$ by structural induction. The other cases for $\mathcal{ALCSCC}^\infty$ are also appearing in $\mathcal{ALCCQU}$; thanks to Corollary 2, we already know that the claim holds for them.

We assume the following inductive hypothesis: for every $\mathcal{ALCSCC}^\infty$ concept description D appearing in k or ℓ and $(e_1, e_2) \in \rho$,

$$e_1 \in D^{\mathcal{I}_1} \text{ if and only if } e_2 \in D^{\mathcal{I}_2}.$$

Let $d_1 \in \mathrm{succ}(k = \ell)^{\mathcal{I}_1}$. Without loss of generality, and adapting Lemma 5 to $\mathcal{ALCSCC}^\infty$ (it can be proved in the same fashion), we assume that

$$k = \sum_{i=1}^{N} N_i \cdot |\omega_i \cap C_i|, \quad \ell = \sum_{j=1}^{m} N'_j \cdot \left|\omega'_j \cap C'_j\right|$$

where ω_i, ω'_j are safe role types, C_i, C'_j are $\mathcal{ALCSCC}^\infty$ concept descriptions and N_i, N'_j are natural numbers.

We show that, under the assumptions made so far, $\left|\omega^{\mathcal{I}_1}(d_1) \cap C^{\mathcal{I}_1}\right| = \left|\omega^{\mathcal{I}_2}(d_2) \cap C^{\mathcal{I}_2}\right|$ for each term $\omega \cap C$ appearing in k and ℓ. By contradiction, assume that $\left|\omega^{\mathcal{I}_1}(d_1) \cap C^{\mathcal{I}_1}\right| \neq \left|\omega^{\mathcal{I}_2}(d_2) \cap C^{\mathcal{I}_2}\right|$; without loss of generality, we assume that

$$\left|\omega^{\mathcal{I}_1}(d_1) \cap C^{\mathcal{I}_1}\right| < \left|\omega^{\mathcal{I}_2}(d_2) \cap C^{\mathcal{I}_2}\right|, \text{ with } \left|\omega^{\mathcal{I}_1}(d_1) \cap C^{\mathcal{I}_1}\right| = N \text{ and } N \in \mathbb{N};$$

under our previous assumption, this is the only possible case, because $\left|\omega^{\mathcal{I}_1}(d_1) \cap C^{\mathcal{I}_1}\right| = \infty$ would imply $\left|\omega^{\mathcal{I}_2}(d_2) \cap C^{\mathcal{I}_2}\right| = \infty$.

Let D_2 be a finite subset of $\omega^{\mathcal{I}_2}(d_2) \cap C^{\mathcal{I}_2} \subseteq \omega^{\mathcal{I}_2}(d_2)$ satisfying $|D_2| = N + 1$. By $\mathcal{ALCQ}t$-bisimilarity of d_1 and d_2 and our inductive hypothesis (similar to the argument used in Theorem 7 for qualified number restrictions), it follows that there exists a finite set $D_1 \subseteq \omega^{\mathcal{I}_1}(d_1) \cap C^{\mathcal{I}_1}$ in bijection with D_2. This leads to a contradiction, since $N + 1 > N$.

Thanks to what we have just shown, it follows that

$$k^{\mathcal{I}_1 d_1} = \sum_{i=1}^{N} N_i \cdot \left|\omega_i^{\mathcal{I}_1}(d_1) \cap C_i^{\mathcal{I}_1}\right| = \sum_{i=1}^{N} N_i \cdot \left|\omega_i^{\mathcal{I}_2}(d_2) \cap C_i^{\mathcal{I}_2}\right| = k^{\mathcal{I}_2 d_2},$$
$$\ell^{\mathcal{I}_1 d_1} = \sum_{j=1}^{m} N'_j \cdot \left|\omega'^{\mathcal{I}_1}_j(d_1) \cap C'^{\mathcal{I}_1}_j\right| = \sum_{j=1}^{m} N'_j \cdot \left|\omega'^{\mathcal{I}_2}_j(d_2) \cap C'^{\mathcal{I}_2}_j\right| = \ell^{\mathcal{I}_2 d_2}$$
$$k^{\mathcal{I}_2 d_2} = k^{\mathcal{I}_1 d_1} = \ell^{\mathcal{I}_1 d_1} = \ell^{\mathcal{I}_2 d_2}.$$

Thus, $d_2 \in \mathrm{succ}(k = \ell)^{\mathcal{I}_2}$. In a similar way, we can prove that the claim holds for the cases $\mathrm{succ}(k \leq \ell)$ and $\mathrm{succ}(k \geq \ell)$. □

In Theorem 13 we showed that the $\mathcal{ALCSCC}^\infty$ concept description $C := \mathrm{succ}(|r \cap A| = |r \cap \neg A|)$ is not expressible in $\mathcal{ALCCQU}$. However, as a consequence of Theorem 14, the concept C is invariant under $\sim_{\mathcal{ALCQ}t}$. Thus, C cannot be expressed as a first-order formula — this is due to Theorem 12.

Corollary 5. *There are $\mathcal{ALCSCC}^\infty$ concept descriptions that cannot be expressed in first-order logic.*

This result paves the road for the final result of this chapter, that shows how the DL $\mathcal{ALCCQU}$ corresponds to a very specific subset of $\mathcal{ALCSCC}^\infty$, that is expressible in first-order logic and that strengthens its definition.

Theorem 15. *$\mathcal{ALCCQU}$ is the first-order fragment of $\mathcal{ALCSCC}^\infty$.*

Proof. Let C be a $\mathcal{ALCSCC}^\infty$ concept description. If C is not expressible in first-order logic, then it does not belong to $\mathcal{ALCCQU}$: we have shown that if C is a $\mathcal{ALCQ}t$ concept description (equivalently, a $\mathcal{ALCCQU}$ concept description), then it is expressible in first-order logic. On the other hand, if C is expressible in first-order logic, from Theorem 12 follows that there exists a $\mathcal{ALCCQU}$ concept description C' that is equivalent to C, thus C is expressible in $\mathcal{ALCCQU}$. □

We have shown how $\mathcal{ALCCQU}$ can be defined using a logical characterization in terms of $\mathcal{ALCSCC}^\infty$. In the next chapter, we focus on the concept satisfiability problem for $\mathcal{ALCCQU}$ without a TBox and we propose a practical algorithm to implement a decision procedure for this problem.

4 Concept satisfiability in $\mathcal{ALCCQU}$

We recall that in principle one could adapt the PSpace algorithm for $\mathcal{ALCSCC}$ concept satisfiability without a TBox shown in [2] to obtain a decision procedure for $\mathcal{ALCCQU}$ concept satisfiability in PSpace However, we already stressed the fact that the mentioned algorithm is practically inefficient: indeed, many of its steps involve non-deterministically guessing, for instance in finding truth assignments for the propositional formula $\mathrm{prop}(C)$.

The aim of this chapter is to devise an algorithm to check $\mathcal{ALCCQU}$ concept satisfiability that replaces the non-deterministic guessing of [2] with techniques borrowed from the fields of SAT solving and ILP.

From $\mathcal{ALCCQU}$ concept descriptions to propositional formulae. A subconcept of a $\mathcal{ALCCQU}$ concept description C is called an $\mathcal{ALCCQU}$*-atom* if it is either a concept name or a role successor constraint. Given a $\mathcal{ALCCQU}$ concept description C, we denote with $\mathrm{prop}(C)$ the propositional formula obtained by replacing every $\mathcal{ALCCQU}$-atom with a propositional variable and the propositional connectives with the ones of $\mathcal{ALCCQU}$. We can define the mapping prop in a recursive way, as follows:

$$\mathrm{prop}(A) := x_A \text{ if } A \text{ is a } \mathcal{ALCCQU}\text{-atom} \qquad \mathrm{prop}(\neg C) := \neg\,\mathrm{prop}(C)$$
$$\mathrm{prop}(C \sqcap D) := \mathrm{prop}(C) \wedge \mathrm{prop}(D) \qquad \mathrm{prop}(C \sqcup D) := \mathrm{prop}(C) \vee \mathrm{prop}(D).$$

Proposition 1. *If the $\mathcal{ALCCQU}$ concept description C is satisfiable, then the propositional formula $\mathrm{prop}(C)$ is satisfiable.*

Proof. Assume that C is satisfiable and let $\mathcal{I}$ be an interpretation such that $C^{\mathcal{I}} \neq \emptyset$. Let μ be the truth assignment defined as follows:

$$\mu(x_A) := 1 \text{ iff } A^{\mathcal{I}} \neq \emptyset \text{ for all } \mathcal{ALCCQU}\text{-atoms } A.$$

Then, the truth assignment μ satisfies $\mu \models \mathrm{prop}(C)$. □

While the encoding of C into $\mathrm{prop}(C)$ yields a complete method to check for $\mathcal{ALCCQU}$ concept satisfiability, the presence of role successor constraints prevents it from enjoying soundness, because of the nesting of concept descriptions inside role successor constraints that is not taken into account. This implies that by itself, converting the $\mathcal{ALCCQU}$ concept description C to a propositional formula $\mathrm{prop}(C)$ is not sufficient to check whether C is satisfiable, as shown in the next example.

Example 7. Let $C := \mathrm{succ}(|A \sqcap \neg A| \geq 1)$ with A an arbitrary concept name. Let x be the propositional formula associated with the role successor constraint in C (up to renaming). The

propositional formula $\mathrm{prop}(C) = x$ is satisfiable; however, C is not satisfiable, because no role successor of C belongs to the interpretation of $A \sqcap \neg A \equiv \bot$.

Proposition 2. *Assume that the $\mathcal{ALCCQU}$ concept description C has no role successor constraint as a subconcept. The following holds:*

$$\textit{if } \mathrm{prop}(C) \textit{ is satisfiable, then } C \textit{ is satisfiable.}$$

Proof. Trivial. □

The design of the algorithm for $\mathcal{ALCCQU}$ concept satisfiability proposed in this chapter abstracts from the implementational details of the invoked SAT solver. We only require that the employed SAT-solver provides a correct decision procedure for propositional satisfiability and that if the input formula is satisfiable the solver returns a model for it. Hereafter, we introduce the procedure `GetSATModel`(φ, S) that calls the chosen SAT solver to decide whether the propositional formula φ is satisfiable; if there is a model μ of φ that is not included in S, the routine returns μ, otherwise it returns `NIL`.

To overcome the issue shown in Example 7, we need to devise a procedure that considers the content of the role successor constraints appearing in C. To this end, we resort to a decidable first-order fragment that admits counting quantifiers, called CQU [11], and we show how to transform role successor constraints into a set of formulae of CQU. In this way, we can then use known techniques to check that such an instance is satisfiable, therefore solving the problem posed by nested $\mathcal{ALCCQU}$ concept descriptions.

4.1 The first-order fragment CQU

In this section, we introduce the syntax and the semantics of a function-free, first-order fragment called *counting quantifiers over unary predicates* (CQU) and first presented in [11]. After that, we illustrate an approach to reduce $\mathcal{ALCCQU}$ concept satisfiability to CQU formula satisfiability with additional checks

Syntax and semantics of CQU. Given a countable set $\mathcal{R}_1$ of unary predicate symbols, a *counting sentence* is a formula of the form $\exists_{\leq N} x.\varphi(x)$ or $\exists_{\geq N} x.\varphi(x)$, where $N \in \mathbb{N}$ and $\varphi(x)$ is a Boolean combination of symbols of $\mathcal{R}_1$; a *universal sentence* has the form $\forall x.\varphi(x)$, where $\varphi(x)$ is restricted as in the case of counting sentences. We say that ψ is a formula in the language of *counting quantifiers over unary predicates* (CQU) if it is a finite conjunction of counting and universal sentences; if $\mathcal{Q}$ and $\mathcal{U}$ are the sets of counting and universal sentences appearing in ψ, respectively, we use the notation $\psi = \langle \mathcal{Q}, \mathcal{U} \rangle$.

An interpretation $\Delta^{\mathcal{I}}$ consists of a non-empty set $\Delta^{\mathcal{I}}$ and a mapping $\cdot^{\mathcal{I}}$ that maps each unary predicate p to a set $p^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$. We denote the variable assignment of x to the value $d \in \Delta^{\mathcal{I}}$ with $\{x/d\}$ We define the satisfiability of a CQU formula under an interpretation $\mathcal{I}$ as follows:

$$
\begin{aligned}
\mathcal{I}, \{x/d\} &\models p(x) && \text{iff } x^{\mathcal{I}} \in p^{\mathcal{I}} \\
\mathcal{I}, \{x/d\} &\models \neg\psi && \text{iff } \mathcal{I}, \{x/d\} \not\models \psi \\
\mathcal{I}, \{x/d\} &\models \psi_1 \wedge \psi_2 && \text{iff } \mathcal{I}, \{x/d\} \models \psi_1 \text{ and } \mathcal{I}, \{x/d\} \models \psi_2 \\
\mathcal{I} &\models \exists_{\geq N} x.\varphi && \text{iff } \left|\{d \in \Delta^{\mathcal{I}} \mid \mathcal{I}, \{x/d\} \models \varphi\}\right| \geq N \\
\mathcal{I} &\models \exists_{\leq N} x.\varphi && \text{iff } \left|\{d \in \Delta^{\mathcal{I}} \mid \mathcal{I}, \{x/d\} \models \varphi\}\right| \leq N \\
\mathcal{I} &\models \forall x.\varphi && \text{iff } \mathcal{I}, \{x/d\} \models \varphi \text{ for all } d \in \Delta^{\mathcal{I}}.
\end{aligned}
$$

We omit the cases for the other Boolean connectives $\vee, \rightarrow, \leftrightarrow$ that are defined as usual. A CQU formula ψ is *satisfiable* if there exists an interpretation $\mathcal{I}$ such that $\mathcal{I} \models \psi$. A set of CQU formulae Γ is satisfiable, if there is an interpretation $\mathcal{I}$ such that $\mathcal{I} \models \gamma$ for each $\gamma \in \Gamma$.

Semantic equivalences in CQU. As a result of the definition of CQU semantics, there are constructors that are not explicitly mentioned in the language syntax but that can be used in building CQU formulae. In particular, the following semantic equivalences hold in CQU:

$$\exists x.\psi \equiv \exists_{\geq 0} x.\psi \tag{4.1}$$

$$\exists_{\leq N} x.\psi \equiv \neg\exists_{\geq N+1} x.\psi \tag{4.2}$$

$$\exists_{=N} x.\psi \equiv \exists_{\geq N} x.\psi \wedge \exists_{\leq N} x.\psi \tag{4.3}$$

$$\forall x.\psi \equiv \exists_{\leq 0} x.\neg\psi \tag{4.4}$$

A normal form for CQU. Let $\psi = \langle \mathcal{Q}, \mathcal{U} \rangle$ be a CQU formula. We say that ψ is in *normal form* if every counting sequence in $\mathcal{Q}$ is of the form $\exists_{\leq N} x.p(x)$ or $\exists_{\geq N} x.p(x)$ with p an atomic unary predicate.

The next lemma shows that it is always possible to reason about the normal form of a CQU formula. Indeed, every CQU formula has an equisatisfiable CQU normal form; the two formulas, however, are not equivalent because the transformation adds fresh predicate symbols that are absent in the original CQU formula.

Lemma 8. *For every CQU formula $\psi = \langle \mathcal{Q}, \mathcal{U} \rangle$ there exists a CQU formula $\psi' = \langle \mathcal{Q}', \mathcal{U}' \rangle$ that is equisatisfiable and in normal form. Such a formula can be obtained from ψ in polynomial time.*

Proof. The proof is adapted from [11]. We start by adding all the universal sentences of ψ to ψ'. Let $\exists_{\bowtie N} x.\varphi(x)$ be a counting sentence in $\mathcal{Q}$. We replace $\varphi(x)$ with

$$\exists_{\bowtie N} x.p'(x) \text{ and } \forall x.(p'(x) \leftrightarrow \varphi(x))$$

in ψ', where p' is a fresh unary predicate symbol. Clearly, if $\exists_{\bowtie N} x.\varphi(x)$ is satisfiable under $\mathcal{I}$, then $\exists_{\bowtie N} x.p'(x)$ and $\forall x.(p'(x) \leftrightarrow \varphi(x))$ are both satisfiable under $\mathcal{I}'$, where $\Delta^{\mathcal{I}'} = \Delta^{\mathcal{I}}$ and $p'^{\mathcal{I}'} = \varphi^{\mathcal{I}}$. Similarly, if $\exists_{\bowtie N} x.p'(x)$ and $\forall x.(p'(x) \leftrightarrow \varphi(x))$ are satisfiable under $\mathcal{I}$, then so is

$\exists_{\bowtie N} x.\varphi(x)$. The resulting formula ψ' is in normal form and has at most twice as many sentences as ψ. □

Example 8. Let $\psi := \langle\{\exists_{\geq 4} x.p(x) \wedge q(x), \exists_{\leq 3} x.\neg p(x)\}, \emptyset\rangle$ be a CQU formula. Applying the transformation described in the proof of Lemma 8, we obtain its CQU normal form

$$\psi' = \langle\{\exists_{\geq 4} x.p_1(x), , \exists_{\leq 3} x.p_2(x)\}, \{\forall x.p_1(x) \leftrightarrow p(x) \wedge q(x), \forall x.p_2(x) \leftrightarrow \neg p(x)\}\rangle.$$

Let $\psi = \langle \mathcal{Q}, \mathcal{U}\rangle$ be a CQU formula in normal form. Then, assuming that $|\mathcal{Q}| = k$, there are k unary predicates $p_1, \ldots, p_k$ that are quantified in $\mathcal{Q}$ (not necessarily distinct). An *elementary term* over $\mathcal{Q}$ has the form $e(x) := \bigwedge_{i=1}^{k} \lambda_i(x)$, where $\lambda_i(x)$ corresponds either to $p_i(x)$ or $\neg p_i(x)$. An elementary term $e(x)$ is *coherent* if the set of CQU formulae $\{\exists x.e(x)\} \cup \mathcal{U}$ is satisfiable.

Example 9. Let $\psi' = \langle \mathcal{Q}, \mathcal{U}\rangle$ be the CQU formula in normal form obtained in Example 8. Then, the elementary term $e(x) := p_1(x) \wedge p_2(x)$ is not coherent, since $\{\exists x.e(x)\} \cup \mathcal{U}$ entails the formula $\exists x.p(x) \wedge q(x) \wedge \neg p(x)$. On the other hand, the elementary term $e'(x) := \neg p_1(x) \wedge p_2(x)$ is coherent: given the interpretation $\mathcal{I}$ where $p^{\mathcal{I}} = \{d\}$ and $q^{\mathcal{I}} = \{e\}$, it holds that $\mathcal{I}, \{x/d\} \models \{\exists x.e'(x)\} \cup \mathcal{U}$.

Since any interpretation that satisfies a formula $\psi = \langle \mathcal{Q}, \mathcal{U}\rangle$ in CQU normal form must satisfy $\mathcal{U}$, it follows that in every model of ψ only coherent elementary terms are interpreted as non-empty subsets of $\mathcal{I}$. If we assumed that a model of ψ also satisfied a non-coherent term, we would obtain a contradiction: if the non-coherent term e is assigned to a non-empty interpretation, it is implied, by definition of coherent terms, that $\mathcal{U}$ is unsatisfiable.

From CQU satisfiability to Linear Algebra. Given an ordering over the sentences of $\mathcal{Q}$, we can encode each elementary term $e(x)$ over $\mathcal{Q}$ as a binary vector $\mathbf{v} = (v_1, \ldots, v_k)$ where $v_i = 1$ if $\lambda_i = p_i$ and $v_i = 0$ if $\lambda_i = \neg p_i$. Let $k_m \leq 2^k$ be the number of coherent elementary terms over $\mathcal{Q}$. We encode all the coherent elementary terms in a matrix A of size $k \times k_m$ of the form

$$A = \left[\mathbf{v}^{(1)} \mid \cdots \mid \mathbf{v}^{(k_m)}\right] = \begin{bmatrix} v_1^{(1)} & \cdots & v_1^{(k_m)} \\ \vdots & \ddots & \vdots \\ v_k^{(1)} & \cdots & v_k^{(k_m)} \end{bmatrix}$$

where the i-th row $\mathbf{a}_i$ of A represents the i-th counting sequence $\exists_{\bowtie_i b_i} x.p_i(x)$ in $\mathcal{Q}$ and the j-th column $A_j = \mathbf{v}^{(j)}$ a coherent elementary term over $\mathcal{Q}$. If we let $\mathbf{b} := (b_1, \ldots, b_k)'$ then the linear system

$$\begin{aligned} &A \cdot \mathbf{x} \bowtie \mathbf{b} \\ &\quad \mathbf{x} \geq \mathbf{0} \text{ and } \mathbf{x} = (x_1, \ldots, x_{k_m}) \text{ is an integer vector} \end{aligned} \tag{4.5}$$

represents a reduction of the original CQU formula to solving a linear system.

Example 10. Let ψ' be the CQU formula in normal form obtained in Example 8. Then, the associated linear system corresponds to

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \begin{matrix} \geq \\ \leq \end{matrix} \begin{bmatrix} 4 \\ 3 \end{bmatrix}$$

with feasible solution $x_1 = 4$, $x_2 = 3$ and $x_3 = 0$. We define an interpretation $\mathcal{I}$ such that $p_1^{\mathcal{I}} = \{d_1, d_2, d_3, d_4\}$ and $p_2^{\mathcal{I}} = \{d_5, d_6, d_7\}$. Then, $\mathcal{I} \models \psi'$.

The example above shows that, given a solution to the linear system associated to a CQU formula in normal form, it is possible to instantiate a satisfying interpretation for the formula. Indeed, the following result holds.

Theorem 16. *A formula $\psi = \langle \mathcal{Q}, \mathcal{U} \rangle$ in CQU normal form is satisfiable if and only if the corresponding linear system* (4.5) *has a solution.*

Proof. Omitted. The proof can be found in [11]. □

4.2 Column generation with SAT oracle

In this section, we show how to integrate a technique commonly employed in integer linear programming called *column generation* [6] in order to decide satisfiability of a CQU formula as done in [11]. Our goal is to use this approach the replace the non-deterministic guessing of the polynomially many Venn regions that should be interpreted as non-empty used in in the algorithm proposed in [2]. Additional information and details about the column generation technique can be found in [6, 10, 17].

In fact, the linear system (4.5) can be seen as an instance of an integer linear program (2.1), where we lift the optimality condition for the solution and we look just for feasibility. Let $\psi = \langle \mathcal{Q}, \mathcal{U} \rangle$ be a CQU formula in normal form. The matrix A obtained in (4.5) contains all the columns associated to coherent elementary terms over the unary predicate $p_1, \ldots, p_k$ occurring in $\mathcal{Q}$.

A possible way to generate these columns is to transform each universal sentence $\mathcal{U}$ into a conjunction of propositional formulae $\mathrm{prop}(\mathcal{U})$ by deleting the universal quantifiers and replacing each unary predicate $p(x)$ in u with a propositional variable p. Using a SAT solver, we can then enumerate all the models of $\mathrm{prop}(\mathcal{U})$ and extract from each of these a column $\mathbf{v}$ containing the values $\mu(p_1), \ldots, \mu(p_k)$ that corresponds to a coherent elementary term. A column of A obtained in this way is called *$\mathcal{U}$-satisfying*.

Example 11. Let $\psi' = \langle \mathcal{Q}, \mathcal{U} \rangle$ be the CQU formula in normal form shown in Example 8. In this case,

$$\mathrm{prop}(\mathcal{U}) = (p_1 \leftrightarrow p \wedge q) \wedge (p_2 \leftrightarrow \neg p).$$

The column $(1, 1)$ is not $\mathcal{U}$-satisfying: if μ was a valuation of $\mathrm{prop}(\mathcal{U})$ such that $\mu(p_1) = \mu(p_2) = 1$, then both $\mu(p) = 1$ and $\mu(p) = 0$ would hold. All the other columns are $\mathcal{U}$-satisfying: for example, $(1, 0)$ can be obtained with the truth assignment μ satisfying $\mu(p) = \mu(q) = 1$.

Lemma 9. *An elementary term e over $\mathcal{Q}$ is coherent if and only if the corresponding vector $\mathbf{v}$ is $\mathcal{U}$-satisfying.*

Proof. Trivial. □

Each step of the generation of A requires invoking a SAT solver; moreover, with each step the input given to the solver increases, since we need to keep track of which columns have already been generated. Finally, A can be exponentially large in the size of $\mathcal{Q}$, since the number of coherent elementary terms over $\mathcal{Q}$ can be 2^k in the worst case, thus creating a very large linear system that needs to be solved.

The intuition behind *column generation* [10] is that it is not always the case that we need all the information contained in A in order to solve our problem; we can rather focus on a subset of the elementary terms, forming a restricted problem and adding new coherent elementary terms in an incremental fashion.

4.2.1 Column generation and CQU

Given a CQU formula $\psi = \langle \mathcal{Q}, \mathcal{U} \rangle$ in normal form and using (4.5), the *master problem* associated with ψ is a primal problem

$$\begin{aligned} &\text{minimize } \mathbf{c}' \cdot \mathbf{x} \\ &\text{subject to } A \cdot \mathbf{x} \bowtie \mathbf{b} \text{ and } \mathbf{x} \geq \mathbf{0} \end{aligned} \tag{4.6}$$

where A is a $k \times 2^k$ matrix containing all the possible valuations of propositional formulae induced by the elementary terms corresponding to $\mathcal{Q}$ and the cost vector $\mathbf{c}$ is given by the following function: for $j = 1, \ldots, k_m$, $c_j = 0$ if and only if the column A_j is $\mathcal{U}$-satisfying.

Example 12. The master problem generated by the formula ψ' obtained in Example 8 is

$$\text{minimize } x_1$$
$$\text{subject to } \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \begin{matrix} \geq \\ \leq \end{matrix} \begin{bmatrix} 4 \\ 3 \end{bmatrix} \quad x_i \geq 0, i = 1, \ldots, 4.$$

We notice that $\mathbf{c}' \cdot \mathbf{x} \geq 0$, since both $\mathbf{c}$ and $\mathbf{x}$ are non-negative vectors. Columns of A that are not $\mathcal{U}$-satisfying should be interpreted as empty sets under each model of ψ: therefore, if A_j is not $\mathcal{U}$-satisfying, we search for solutions that satisfy $x_j = 0$. Thus, we only seek feasible solutions for the problem

$$\begin{cases} \mathbf{c}' \cdot \mathbf{x} = 0 \\ A \cdot \mathbf{x} \bowtie \mathbf{b} \\ \mathbf{x} \geq \mathbf{0} \end{cases} \tag{4.7}$$

To avoid the explicit generation of the whole matrix A unless needed, we start the search for a solution by focusing on a *restricted master problem*

$$\begin{aligned} &\text{minimize } \mathbf{c}'_r \cdot \mathbf{x}_r \\ &\text{subject to } A_r \cdot \mathbf{x}_r \bowtie \mathbf{b} \text{ and } \mathbf{x}_r \geq \mathbf{0} \end{aligned} \tag{4.8}$$

where A_r is a matrix extracted from A with the same number of rows — adhering to [11] we set $A_r := I_k$ — and $\mathbf{c}_r$ is computed over the columns of A_r as described above. If (4.8) is unfeasible, then the master problem (4.7) is unfeasible as well. Assume that (4.8) has a feasible solution x_r. If $\mathbf{c}'_r \cdot \mathbf{x}_r = 0$, then the vector $\mathbf{x}$ where every variable not appearing in x_r is set to 0 is a solution for (4.7). Otherwise, we search if it is possible to decrease the objective function of (4.8) by adding a column of A to A_r.

Example 13. The restricted problem obtained from the master problem in Example 12 is

$$\begin{aligned} &\text{minimize } 0 \\ &\text{subject to } \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_2 \\ x_3 \end{bmatrix} \begin{matrix} \geq \\ \leq \end{matrix} \begin{bmatrix} 4 \\ 3 \end{bmatrix} \quad x_2, x_3 \geq 0. \end{aligned}$$

In this case, the objective function has the constant value 0. Therefore, any integral solution to the restricted master problem yields a solution to the original problem. For example, the solution $x_2 = 4$ and $x_3 = 3$ can be extended to a solution of the master problem in Example 12 by setting $x_1 = x_4 = 0$.

In column generation, the condition for searching for columns of A to be added to A_r is borrowed from the one used in the *simplex method* [7] and requires the presence of a dual solution to (4.8). If the restricted problem has no dual solution, the column generation procedure *fails*. Assume that (4.8) has a feasible dual solution $\mathbf{z}_r$. The problem of choosing a column from A that decreases the objective function of (4.8) when added to A_r can be stated by introducing the *reduced cost* function of adding the j-th column of A to A_r, which is

$$c_j^\star := c_j - \mathbf{z}_r \cdot A_j. \tag{4.9}$$

A known result from the field of linear optimization is that if $c_j^\star \geq 0$ for all columns A_j, then the solution $\mathbf{x}$ is optimal for both problems (4.6) and (4.8) [10]. Therefore, choosing a column from A that added to A_r decreases $\mathbf{c}'_r \cdot \mathbf{x}_r$ amounts fo find a feasible solution to the linear inequality

$$c_j - \mathbf{z}'_r \cdot A_j < 0.$$

We are interested in feasibility of the master problem, rather than in finding an optimal solution; thus, we aim only at obtaining a non-increasing value of the objective function. Hence, we can adapt the linear inequality to obtain

$$c_j - \mathbf{z}'_r \cdot A_j \leq 0.$$

We are only interested adding columns of A that are $\mathcal{U}$-satisfying — this implies that $c_j = 0$ holds. Thus, we are looking for a solution to the linear inequality

$$-\mathbf{z}_r \cdot \mathbf{v} \leq 0, \quad \mathbf{v} \in \{0,1\}^n. \tag{4.10}$$

If (4.10) has no solution, then the objective function of (4.6) cannot be decreased any further. If a feasible solution $\mathbf{v}$ for (4.10) exists, then adding it as a column to A_r might decrease the value of the objective function or maintain its value, but not increasing it.

By appending the column $\mathbf{v}$ to A_r and 0 to $\mathbf{c}_r$, we obtain another instance of the restricted problem (4.8) with smaller or equal objective function. We can then reiterate the procedure of column generation until one of the possible outcomes is obtained:

1. we obtain a primal solution of (4.8) that satisfies $\mathbf{c}'_r \cdot \mathbf{x}_r = 0$;

2. either the primal/dual solutions of (4.8) or the solution of (4.10) are not available; in this case, problem (4.7) is unfeasible;

3. all the columns of A have been added, and we check whether $\mathbf{c}'_r \cdot \mathbf{x}_r = \mathbf{c}' \cdot \mathbf{x} = 0$.

Since A is a finite matrix, the column generation algorithm is exact [10]. However, to ensure termination, we need to check that the generated column is not already appearing in A_r: this is a consequence of relaxing the inequality (4.10) to be non-strict. Indeed, in the strict case, every column A_j of A is added to A_r at most once, since no variable in an optimal restricted master problem has negative reduced cost $c^\star_j$ [10].

Example 14. Let

$$\psi' = \langle \{\exists_{\geq 4} x.p_1(x), , \exists_{\leq 3} x.p_2(x)\}, \{\forall x.p_1(x) \leftrightarrow p(x) \wedge q(x), \forall x.p_2(x) \leftrightarrow p(x)\}\rangle$$

be a CQU formula in normal form. The restricted master problem for ψ' is initialized to

$$\begin{aligned} &\text{minimize } x_1 \\ &\text{subject to } \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \begin{matrix} \geq \\ \leq \end{matrix} \begin{bmatrix} 4 \\ 3 \end{bmatrix} \quad x_1, x_2 \geq 0. \end{aligned}$$

A feasible solution for this problem is $x_1 = 4$ and $x_2 = 3$; however, $x_1 \neq 0$, therefore we try to reduce the objective function x_1 by adding a column. In order to generate a column to be added to the restricted master problem, if possible, we solve the dual problem

$$\begin{aligned} &\text{maximize } 4z_1 + 3z_2 \\ &\text{subject to } \begin{bmatrix} z_1 & z_2 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{matrix} \leq \\ \leq \end{matrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \qquad z_1 \leq 0, z_2 \geq 0. \end{aligned}$$

which has a feasible and integral solution $z_1 = z_2 = 0$ (in this case, it is also optimal). Since the yielded inequality is $0 \leq 0$, which has all the possible $\mathcal{U}$-satisfying columns as solutions, we add the column $(1, 1)$ to the restricted master problem:

$$\begin{aligned} &\text{minimize } x_1 \\ &\text{subject to } \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \begin{matrix} \geq \\ \leq \end{matrix} \begin{bmatrix} 4 \\ 3 \end{bmatrix} \quad x_1, x_2, x_3 \geq 0. \end{aligned}$$

We obtain a feasible solution $x_1 = 4$, $x_2 = 3$ and $x_3 = 0$ with $x_1 \neq 0$. The new dual problem

$$\begin{aligned} &\text{maximize } 4z_1 + 3z_2 \\ &\text{subject to } \begin{bmatrix} z_1 & z_2 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix} \begin{matrix} \leq \\ \leq \\ \leq \end{matrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \qquad z_1 \leq 0, z_2 \geq 0 \end{aligned}$$

yields the same solution $z_1 = z_2 = 0$ as in the previous iteration, which leads to the inequality $0 \leq 0$. We add to the restricted problem the only remaining column $(0, 0)$, obtaining

$$\begin{aligned} &\text{minimize } x_1 \\ &\text{subject to } \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \begin{matrix} \geq \\ \leq \end{matrix} \begin{bmatrix} 4 \\ 3 \end{bmatrix} \quad x_1, x_2, x_3, x_4 \geq 0. \end{aligned}$$

This problem has no solution such that $x_1 = 0$. Indeed, the formula ψ' is unsatisfiable.

4.2.2 From linear inequalities to propositional formulae

Each column of A in (4.7) is a binary vector: using this information, we can use a SAT solver to generate the $\mathcal{U}$-satisfying columns of A, instead of explicitly enumerating them. This implicit enumeration is a clear advantage, since we only generate all the columns of A in the worst-case. In this paragraph, we show how to reduce the problem of finding a feasible solution to (4.10) to satisfiability of a propositional formula. The reduction that we use is an adaptation of the linear encoding proposed in [21].

Hereafter, let $S(L, U)$ denote the sum $\sum_{i=L}^{U} -z_i v_i$, with $1 \leq L \leq U \leq n$. Let $M := \lceil \log_2(z^\star + 1) \rceil$ with $z^\star := \max_{1 \leq i \leq n} |z_i|$ be the number of bits necessary to represent each component z_i of $\mathbf{z}$ in binary notation and B_i the set containing the positions of the binary representation of $|z_i|$ that are equal to 1. For each component v_i of $\mathbf{v}$ we introduce a propositional decision variable x_i. To encode the k-th digit of the binary representation of $S(L, U)$, we introduce a propositional variable $s_k^{(L,U)}$, with $0 \leq k \leq M_L^U$ and $M_L^U := M + \log_2(U - L + 1)$; when $L = U$, $M_L^U = M$ holds.

We map the sum $S(L,U)$ to a propositional formula by means of the function σ, recursively defined as follows:

$$\sigma(S(L,U)) = \begin{cases} \sigma(S(L, \left\lfloor \frac{L+U}{2} \right\rfloor)) \wedge \sigma(S(\left\lfloor \frac{L+U}{2} \right\rfloor + 1), U) \wedge T_\Sigma(L,U) & L < U \\ \bigwedge\limits_{\substack{k \in B_L \\ z_L < 0}} (s_k^{(L,U)} \leftrightarrow x_L) \wedge \bigwedge\limits_{\substack{k \in B_L \\ z_L \geq 0}} (s_k^{(L,U)} \leftrightarrow \neg x_L) \wedge \bigwedge\limits_{k \notin B_L} \neg s_k^{(L,U)} & L = U \end{cases}$$

where the propositional formula $T_\Sigma(L,U)$ encodes the sum of $S(L, \lfloor \frac{L+U}{2} \rfloor)$ and $S(\lfloor \frac{L+U}{2} \rfloor + 1), U)$. In the definition of $T_\Sigma(L,U)$ we introduce auxiliary propositional variables $c_{i,i+1}^{(L,U)}$ for $1 \leq i < M_L^U$ encoding the *carrying* done during the addition:

$$\begin{aligned} C_0^{(L,U)} &:= \left(c_{0,1}^{(L,U)} \leftrightarrow \left(s_0^{(L, \lfloor \frac{L+U}{2} \rfloor)} \wedge s_0^{(\lfloor \frac{L+U}{2} \rfloor + 1, U)} \right) \right) \\ C_j^{(L,U)} &:= \Bigg(c_{j,j+1}^{(L,U)} \leftrightarrow \Bigg(\left(s_j^{(L, \lfloor \frac{L+U}{2} \rfloor)} \wedge s_j^{(\lfloor \frac{L+U}{2} \rfloor + 1, U)} \right) \vee \\ &\qquad \vee \left(s_j^{(L, \lfloor \frac{L+U}{2} \rfloor)} \wedge c_{j-1,j}^{(L,U)} \right) \vee \left(s_j^{(\lfloor \frac{L+U}{2} \rfloor + 1, U)} \wedge c_{j-1,j}^{(L,U)} \right) \Bigg) \Bigg) \\ C_{M_L^U}^{(L,U)} &:= \left(c_{M_L^U - 1, M_L^U}^{(L,U)} \leftrightarrow s_{M_L^U}^{(L,U)} \right) \\ D_0^{(L,U)} &:= \left(s_0^{(L,U)} \leftrightarrow \left(s_0^{(L, \lfloor \frac{L+U}{2} \rfloor)} \leftrightarrow \neg s_0^{(\lfloor \frac{L+U}{2} \rfloor + 1, U)} \right) \right) \\ D_j^{(L,U)} &:= \left(s_j^{(L,U)} \leftrightarrow \left(s_j^{(L, \lfloor \frac{L+U}{2} \rfloor)} \leftrightarrow \neg s_j^{(\lfloor \frac{L+U}{2} \rfloor + 1, U)} \leftrightarrow c_{j-1,j}^{(L,U)} \right) \right) \\ T_\Sigma(L,U) &= \textstyle\bigwedge_{j=0}^{M_L^U} \left(D_j^{(L,U)} \wedge C_j^{(L,U)} \right) \end{aligned}$$

To encode the information that $-\mathbf{z} \cdot \mathbf{v}$ is non-positive, we add the conjunct $\bigwedge_{i=1}^{n} \neg s_i^{(1,n)}$.

Theorem 17. *The linear inequality $-\mathbf{z} \cdot \mathbf{v} \geq 0$ has a feasible solution that is $\mathcal{U}$-satisfying (up to matching the decision variables in $\mathbf{v}$ with the variables associated to the counting sentences in $\mathcal{Q}$) if and only if the propositional formula*

$$\sigma(S(1,n)) \wedge \textstyle\bigwedge_{i=1}^{n} \neg s_i^{(1,n)} \wedge \mathrm{prop}(\mathcal{U}) \tag{4.11}$$

is satisfiable.

Proof. See [21] for the correctness of the reduction. Adding $\mathrm{prop}(\mathcal{U})$ to the propositional formula ensures that all of its models yield $\mathcal{U}$-satisfying solutions. □

Example 15. The encoding of the inequality seen in Example 14 is (up to renaming and simplifying)

$$\bigwedge_{i=0}^{2} \neg s_i^{(1,2)} \wedge \neg s_i^{(1,1)} \wedge \neg s_i^{(2,2)} \wedge \bigwedge_{j=0}^{1} \neg c_{j,j+1}^{(1,2)} \wedge (v_1 \leftrightarrow p \wedge q) \wedge (v_2 \leftrightarrow p).$$

The assignment $v_1 = v_2 = 1$ and $v_1 = v_2 = 0$ can be extracted from models of this formula; therefore, it is legitimate to add them as columns to the restricted problem in Example 14 during the column generation phase.

4.2.3 Column generation and $\mathcal{ALCCQU}$

In order to use column generation within the context of $\mathcal{ALCCQU}$ concept satisfiability, we need to tackle additional challenges, such as ensuring that a generated $\mathcal{U}$-satisfying column does not yield an unsatisfiable concept description. After the definition of the encoding of $\mathcal{ALCCQU}$ into CQU, we illustrate how to overcome this issue using an introductory example.

From role successor constraints to CQU formulae. We define a mapping γ_x from $\mathcal{ALCCQU}$ set terms to unary first-order formulas that maps each role name r to a unary predicate $R_r(x)$ and each $\mathcal{ALCCQU}$-atom A to a unary predicate $Q_c(x)$. The mapping γ_x is extended to all $\mathcal{ALCCQU}$ set terms as follows:

$$\begin{aligned}
&\gamma_x(C \sqcap D) := \gamma_x(C) \wedge \gamma_x(D) && \gamma_x(C \sqcup D) := \gamma_x(C) \vee \gamma_x(D)\\
&\gamma_x(\neg C) := \neg\gamma_x(C) && \gamma_x(\mathrm{succ}(|s| \bowtie n)) := Q_s(x)\\
&\gamma_x(\emptyset) := \bigwedge_{r \in N_R} \neg R_r(x) && \gamma_x(\mathcal{U}) := \bigvee_{r \in N_R} R_r(x)\\
&\gamma_x(s \sqcap t) := \gamma_x(s) \wedge \gamma_x(t) && \gamma_x(s \sqcup t) := \gamma_x(s) \vee \gamma_x(t)\\
&\gamma_x(s^c) := \neg\gamma_x(s)
\end{aligned}$$

Encoding a role successor constraint of the form $\mathrm{succ}(|s| \geq n)$ or $\mathrm{succ}(|s| \leq n)$ into a CQU formula $\exists_{\geq n} x.\gamma_x(s)$ or $\exists_{\leq n} x.\gamma_x(s)$ seems a promising solution to check $\mathcal{ALCCQU}$ concept satisfiability.

Theorem 18. *If the $\mathcal{ALCCQU}$ role successor constraints $\mathrm{succ}(|s_i| \bowtie_i n_i)$ are satisfiable for $i = 1, \ldots, m$, then the CQU instance made by the counting sentences $\exists_{\bowtie_i n_i} x.\gamma_x(s_i)$ is satisfiable.*

Proof. Let $\mathcal{I}$ be an interpretation such that $d \in \mathrm{succ}(|s_i| \bowtie_i n_i)^{\mathcal{I}}$ for $i = 1, \ldots, m$. We show that $\mathcal{I}_d \models \exists_{\bowtie_i n_i} x.\gamma_x(s_i)$ for $i = 1, \ldots, m$. Since $d \in \mathrm{succ}(|s_i| \bowtie_i n_i)^{\mathcal{I}}$, there are individuals $d_1^{(i)}, \ldots, d_k^{(i)} \in \Delta^{\mathcal{I}}$ such that $d_1^{(i)}, \ldots, d_{n_i}^{(i)} \in s^{\mathcal{I}_d}$. Using structural induction over the set term s, it is possible to show that $d_j^{(i)} \in s^{\mathcal{I}_d}$ implies $d_j^{(i)} \in \gamma_x(s)^{\mathcal{I}_d}$ for $j = 1, \ldots, n_i$, hence $\mathcal{I}_d, \{x/d_j^{(i)}\} \models \gamma_x(s_i)$ for $j = 1, \ldots, n_i$. This allows us to conclude that $\mathcal{I}_d \models \exists_{\bowtie_i n_i} x.\gamma_x(s_i)$ for $i = 1, \ldots, m$. □

Similarly to what mentioned in [2], we need to take into account the semantics of the role successor constraints in the reduction to a CQU instance. In particular, as already mentioned

in [2], we must ensure that the fact that every individual implied in a the set universe of a solution is a proper role successor. In the original algorithm using QFBAPA formulas, that amounted to adding the set constraints $\mathcal{U} = X_{r_1} \cup \cdots \cup X_{r_n}$ to the obtained formula. In our setting, we add the universal sentence $\forall x.\gamma_x(\mathcal{U})$ to the CQU formula yielded by the encoding of a given $\mathcal{ALCCQU}$ concept description.

Another problem is that further nesting of role successor constraints within one another — a feature that is not present in CQU — can yield an unsatisfiable concept description that is deemed as satisfiable, when encoded into a CQU instance.

Example 16. As an example, let $C := \mathrm{succ}(|\mathrm{succ}(|A \sqcap \neg A| \geq 1)| \geq 2)$ for an arbitrary concept name A. The resulting CQU formula $\exists_{\geq 2} x.Q_{\mathrm{succ}(|A \sqcap \neg A| \geq 1)}(x)$ is clearly satisfiable; however, C is unsatisfiable.

This shows that we need to take into account that a certain unary predicate encodes a role successor constraint, when checking the satisfiability of a CQU instance induced from a $\mathcal{ALCCQU}$ concept description. This amounts to ensure that in the linear system (4.5), we interpret as nonempty predicates only those that are $\mathcal{U}$-satisfying and not yielding unsatisfiable $\mathcal{ALCCQU}$ concepts. Hence, we need to make recursive calls to check $\mathcal{ALCCQU}$ concept satisfiability. We illustrate what this means, using an example.

Example 17. Let $C := \mathrm{succ}(|\mathrm{succ}(|A \sqcap \neg A| \geq 2) \sqcap \neg A| \geq 1) \sqcap \mathrm{succ}(|r \cap A| \leq 3)$. The normal form of the CQU instance ψ induced by C is

$$\mathcal{Q} := \{\exists_{\geq 1} x.p_1(x), \exists_{\leq 3} x.p_2(x)\},$$
$$\mathcal{U} := \{\forall x.p_1(x) \leftrightarrow \neg Q_A(x) \wedge Q_{\mathrm{succ}(|A \sqcap \neg A| \geq 2)}(x), \forall x.p_2(x) \leftrightarrow R_r(x) \wedge Q_A(x)\}.$$

If we initialized the restricted master problem as

$$\begin{aligned} &\text{minimize } 0 \\ &\text{subject to } \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \geq \begin{bmatrix} 1 \\ 3 \end{bmatrix} \quad x_1, x_2 \geq 0 \end{aligned}$$

then $x_1 = 1$ and $x_2 = 3$ would be a solution to the master problem, since the objective function is always zero. However, p_1 — in particular, $Q_{\mathrm{succ}(|A \sqcap \neg A| \geq 2)}$ — cannot be interpreted as a nonempty set, because this would imply that there is an individual belonging to the interpretation of $\mathrm{succ}(|A \sqcap \neg A| \geq 2)$, which is clearly unsatisfiable.

We change the definition of the cost function such that $c_j = 0$ if and only if A_j is $\mathcal{U}$-satisfying and the concept description yielded by the column A_j is satisfiable. In this case, the restricted master problem would be

$$\begin{aligned} &\text{minimize } x_1 \\ &\text{subject to } \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \geq \begin{bmatrix} 1 \\ 3 \end{bmatrix} \quad x_1, x_2 \geq 0 \end{aligned}$$

since $(1, 0)$ is $\mathcal{U}$-satisfying but $C_v := \neg A \sqcap \mathrm{succ}(|A \sqcap \neg A| \geq 2)$ is unsatisfiable.

A dual and integral solution to this problem is $\mathbf{z} = (0, 0)$. Similarly to Example 14 we can add any $\mathcal{U}$-satisfying column to the restricted problem. The column $(1, 1)$ is not $\mathcal{U}$-satisfying. The column $(0, 0)$, on the other hand, is $\mathcal{U}$-satisfying and yields a satisfiable concept description. In particular, we choose the model μ of $\mathrm{prop}(\mathcal{U})$ that satisfies $\mu(\mathrm{succ}(|A \sqcap \neg A| \geq 2)) = 0$ and $\mu(A) = 1$; this yields the concept description $C_v := A \sqcap \neg\,\mathrm{succ}(|A \sqcap \neg A| \geq 2)$, which is satisfiable. Therefore, we can add the column to the problem and proceed to check if there is a solution that brings the objective function to 0.

Let $\psi = \langle \mathcal{Q}, \mathcal{U} \rangle$ be a CQU formula in normal form and let $\mathbf{v}$ be a $\mathcal{U}$-satisfying column. Let μ be a model of $\mathrm{prop}(\mathcal{U})$ that yields the column $\mathbf{v}$. If $\mathtt{atoms}(\mathcal{U})$ is set of $\mathcal{ALCCQU}$-atoms that are encoded in $\mathcal{U}$, the concept description C_v is defined as

$$C_v := \bigsqcap\{A \in \mathtt{atoms}(\mathcal{U}) \mid \mu(Q_A) = 1\} \sqcap \bigsqcap\{\neg A \mid A \in \mathtt{atoms}(\mathcal{U}) \wedge \mu(Q_A) = 0\}.$$

If we combine column generation to solve the integer linear program (4.7) with a recursive concept satisfiability check for each concept C_v yielded by a candidate column, we obtain a method that allows us to correctly check for $\mathcal{ALCCQU}$ concept satisfiability, as shown in the next section.

4.3 Branch-and-Price for $\mathcal{ALCCQU}$ concept satisfiability

The goal of the concept satisfiability decision procedure for $\mathcal{ALCCQU}$ is to find whether a given concept description has a feasible interpretation and not a minimal one. This amounts to find a feasible solution to the linear system (4.5) that is obtained by first inducing a CQU instance ψ in normal form from the input concept description C in $\mathcal{ALCCQU}$ normal form and then transforming ψ into a linear system of inequalities — as we have seen in Section 4.2, we do not explicitly produce the system, but we obtain a solution to it by using column generation to solve (4.7).

The algorithm $\mathcal{ALCCQU}$-BB. Given a $\mathcal{ALCCQU}$ concept description C in $\mathcal{ALCCQU}$ normal form, we devise the algorithm $\mathcal{ALCCQU}$-BB that performs the following steps:

1. Using a SAT solver, we check whether $\mathrm{prop}(C)$ is satisfiable. If so, we get a truth assignment μ for the propositional variables in $\mathrm{prop}(C)$; otherwise, we conclude that C is unsatisfiable — this is true, thanks to Proposition 1.

2. Let S be the set of all the role successor constraints in C whose associated propositional variable is evaluated as true under μ; we map S into a CQU instance $\psi = \langle \mathcal{Q}, \mathcal{U} \rangle$ in normal form, using the encoding described in Theorem 18 and obtaining the normal form as explained in Lemma 8. We add the universal sentence $\forall x.\gamma_x(\mathcal{U})$ to ψ to ensure that all the considered invididuals are proper role successors.

3. We check whether the linear program (4.7) associated to ψ is satisfiable using the *branch-and-price* method [6] — branch-and-bound adapted to the context of CQU [11, 19] with the aid of column generation — and taking care of the possible issues related to $\mathcal{ALCCQU}$ outlined in Section 4.2 If an integral solution is found, then the algorithm returns **true**; otherwise, the truth assignment μ obtained in Step 1 is discarded. The algorithm keeps

Algorithm 1 $\mathcal{ALCCQU}$-BB: $\mathcal{ALCCQU}$ concept satisfiability using branch-and-bound

Input: A $\mathcal{ALCCQU}$ concept description C in normal form
Output: **true** if C is satisfiable, **false** otherwise.

```
1:  noGoods ← ∅
2:  while prop(C) has a model not belonging to noGoods do
3:    μ ← GetSATModel(prop(C), noGoods)
4:    if GetConstraints(C, μ) = ∅ then
5:      return true
6:    else
7:      ψ ← SetCQUInstance(GetConstraints(C, μ))
8:      PbSet ← {ψ}
9:      while PbSet ≠ ∅ do
10:       current ← GetProblem(PbSet)
11:       solution ← SolveRestrictedRelaxation(current)
12:       if solution = NIL then
13:         continue
14:       else if solution is integer then
15:         return true
16:       else
17:         var ← GetBranchVar(solution)
18:         PbSet.add(GetBoundedProblems(current, var))
19:       end if
20:     end while
21:   end if
22:   noGoods.add(μ)
23: end while
24: return false
```

track of the truth assignments for $\mathrm{prop}(C)$ that yield an unsatisfiable CQU instance by storing them in the set $noGoods$, which is initially empty.

A detailed version of the $\mathcal{ALCCQU}$-BB algorithm is given in Algorithm 1. Line 3 corresponds to Step 1; Step 2 is carried out at line 8; the lines 8–20 — highlighted in Algorithm 1 — correspond to Step 3.

Heuristics for branch-and-bound. Once we obtained a CQU instance ψ from S, we employ the *branch-and-bound* technique to solve it [19]. We initialize the set of bounded problems $PbSet$ with $\{\psi\}$. A single iteration of the branch-and-bound method works as follows:

1. The algorithm extracts a problem from $PbSet$, according to the following heuristic implemented in `GetProblem`:
   a) if $PbSet$ contains only one problem, the algorithm simply extracts it;
   b) if the set contains more than one problem, the algorithm selects the one which current solution contains the least number of non-integral components;

c) if there is a tie between problems, the one with the largest number of counting sentences is chosen.

2. The algorithm searches for a relaxed solution to the master problem (4.7) induced from the chosen CQU instance using column generation.

3. If the chosen problem has no feasible solution, the algorithm discards it and proceeds to the next iteration; if it has a feasible integral solution, such a solution also satisfies the original problem, since the solution space of the current problem is always included in that of the original problem.

4. If the chosen problem has a feasible rational solution, then there is at least one component x_i in the current solution $\mathbf{x}$ that is not integral; the branching phase is then deployed by `GetBranchVar` and the heuristic employed to choose the branching variable is to select the component x_i of $\mathbf{x}$ which non-integral value is closest to either $\lfloor x_i \rfloor$ or $\lceil x_i \rceil$.

5. The bounding phase is carried by the procedure `GetBoundedProblems` that takes the current CQU instance $\varphi = \langle \mathcal{Q}, \mathcal{U} \rangle$ and adds to $PbSet$ the bounded problems

$$\varphi' := \langle \mathcal{Q} \cup \{\exists_{\leq \lfloor x_i \rfloor} x.p'(x)\}, \mathcal{U} \cup \{\forall x.p'(x) \leftrightarrow e_i(x)\} \rangle \tag{4.12}$$

$$\varphi'' := \langle \mathcal{Q} \cup \{\exists_{\geq \lceil x_i \rceil} x.p'(x)\}, \mathcal{U} \cup \{\forall x.p'(x) \leftrightarrow e_i(x)\} \rangle \tag{4.13}$$

where $e_i(x)$ is the elementary term corresponding to the i-th column of A.

The heuristic implemented in `GetProblem` is well-defined: in the first iteration of the loop, the set contains only ψ, so it is extracted with no need for a solution; if two or more problems are in the set in the $i+1$-th iteration, they are bounded problems that have been obtained by bounding the search space of previously considered problems that have a feasible solution.

In choosing a LP solver, the only requirement is that the solver is able to provide, when existing, feasible primal and dual solutions (not necessarily optimal ones); we make the assumption that the procedure employed by the solver is correct and terminating — an example of such a procedure is the *simplex method* [7]. Hereafter, we denote with `GetPrimalSolution`$(A, \bowtie, \mathbf{b}, \mathbf{c})$ a call to the chosen LP solver that either returns a feasible relaxed solution to (2.1) or returns `NIL`otherwise; the procedure `GetDualSolution`$(A, \bowtie, \mathbf{b}, \mathbf{c})$ works analogously for (2.2), with the additional condition that the dual solution $\mathbf{z}$ is integral — otherwise, we would not be able to apply the propositional encoding described in Section 4.2.

Solving CQU instances and recursive satisfiability. At each stage of the branch-and-price phase of $\mathcal{ALCCQU}$-BB, we check if the relaxation of the selected CQU instance has a solution via the `SolveRestrictedRelaxation` procedure, described in Algorithm 2. This procedure encodes the process described in Section 4.2, with additional steps to ensure that the generated column is not yielding an unsatisfiable $\mathcal{ALCCQU}$ concept. Assume that $\mathbf{v}$ is a $\mathcal{U}$-satisfying col-

Algorithm 2 `SolveRestrictedRelaxation`(ψ)

Input: A CQU formula $\psi = \langle \mathcal{Q}, \mathcal{U} \rangle$ in normal form with $|\mathcal{Q}| = k$.
Output: A relaxed solution to ψ if it exists; `NIL` otherwise.

$A \leftarrow I_k$ and $\mathbf{c} \leftarrow$ `SetCost`$(A, \mathcal{U})$ and $\mathbf{x} \leftarrow \mathbf{b}$
while $\mathbf{c}' \cdot \mathbf{x} > 0$ **do**
  $\mathbf{z} \leftarrow$ `GetDualSolution`$(A, \bowtie, \mathbf{b}, \mathbf{c})$
  if $\mathbf{z}'A \leq \mathbf{c}, \mathbf{z} \bowtie^{-1} \mathbf{0}$ with $\mathbf{z}$ integral is unfeasible **then**
    return `NIL`
  end if
  $\mathbf{v} \leftarrow$ `GenerateColumn`$(\mathbf{z}, \mathcal{U})$
  if $\mathbf{v} =$ `NIL` **then**
    return `NIL`
  end if
  if $\mathcal{ALCCQU}$-BB(C_v) **then**
    if $\mathbf{v}$ is not yet in A **then**
      $A.append(\mathbf{v})$; $\mathbf{c}.append(0)$
    end if
  else
    $\mathcal{U}.add(\forall x.\neg\gamma_x(C_v))$
  end if
  $\mathbf{x} \leftarrow$ `GetPrimalSolution`$(A, \bowtie, \mathbf{b}, \mathbf{c})$
  if $A\mathbf{x} \bowtie \mathbf{b}, \mathbf{x} \geq \mathbf{0}$ is unfeasible **then**
    return `NIL`
  end if
end while
return $\mathbf{x}$

umn obtained as a solution of (4.11) or a column of I_k and let μ be the truth assignment over (4.11) generated by the SAT solver. The $\mathcal{ALCCQU}$ concept description induced by μ is obtained as

$$\begin{aligned} C_v := & \bigsqcap \{A \mid A \text{ is a } \mathcal{ALCCQU}\text{-atom}, \ \mu(x_A) = 1\} \sqcap \\ & \sqcap \bigsqcap \{\neg A \mid A \text{ is a } \mathcal{ALCCQU}\text{-atom}, \ \mu(x_A) = 0\}. \end{aligned} \tag{4.14}$$

Once C_v is obtained, the algorithm makes a recursive call to check that C_v is satisfiable. If C_v is unsatisfiable and $\mathbf{v}$ is a column of I_k, we can assign cost 1 to $\mathbf{v}$ and keep it in the generated matrix: if a solution $\mathbf{x}$ with $\mathbf{c}' \cdot \mathbf{x} = 0$ is found, then the value of the variable associated to $\mathbf{v}$ must be 0, thus it is interpreted as an empty set. We denote that procedure that computes the cost function with this additional feature with `SetCost`$(A, \mathcal{U})$.

If $\mathbf{v}$ is not part of I_k and C_v is unsatisfiable, the algorithm does not add $\mathbf{v}$ as a column to A; instead, the universal sentence $\forall x.\neg\gamma_x(C_v)$ is added. This is done because of two motivations:

1. $\mathbf{v}$ cannot be added to A as a column that is not $\mathcal{U}$-satisfying; in generating $\mathbf{v}$ we assumed that its associated cost is 0, whereas its cost as a not $\mathcal{U}$-satisfying column would be 1 according to the cost function used in the algorithm;

2. $\mathbf{v}$ cannot be entirely discarded, because there might be another valuation μ' that makes $\mathbf{v}$ $\mathcal{U}$-satisfying and that yields a satisfiable $\mathcal{ALCCQU}$ concept description.

This step is unnecessary in the simpler setting of CQU, because every atomic predicate has no hidden semantics, whereas $\mathcal{ALCCQU}$-atoms can be nested role successor constraints which meaning is not taken into account.

4.4 Correctness of $\mathcal{ALCCQU}$-BB

Given a set term s, the set $\texttt{atoms}(s)$ contains exactly the $\mathcal{ALCCQU}$-atoms occurring in s. The *constraint depth* of a $\mathcal{ALCCQU}$ concept description C in normal form is recursively defined as follows:

$$\texttt{cd}(C) := \begin{cases} 0 & C = A \in N_C \\ \texttt{cd}(D) & C = \neg D \\ \max(\texttt{cd}(D_1), \texttt{cd}(D_2)) & C = D_1 \star D_2, \star \in \{\sqcap, \sqcup\} \\ 1 + \max\{\texttt{cd}(D) \mid D \in \texttt{atoms}(s)\} & C = \mathrm{succ}(|s| \bowtie n) \end{cases} \tag{4.15}$$

Lemma 10. *If* $\texttt{cd}(C) = 0$*, then*

$$\mathcal{ALCCQU}\text{-}BB(C) = \textbf{\textit{true}} \text{ if and only if } C \text{ is satisfiable.}$$

Moreover, $\mathcal{ALCCQU}$*-BB terminates.*

Proof. Assume that $\texttt{cd}(C) = 0$. It follows that C has no role successor constraint as a subconcept. If the concept C is satisfiable, then $\mu \neq \texttt{NIL}$, with $\mu = \texttt{GetSATModel}(\mathrm{prop}(C), \emptyset)$ a model of $\mathrm{prop}(C)$ — this is a consequence of Proposition 1. Moreover, $\texttt{GetConstraints}(C, \mu) = \emptyset$, because of the absence of role successor constraints in C. Therefore, $\mathcal{ALCCQU}$-BB(C) returns **true**.

If the concept C is unsatisfiable, then Proposition 2 yields the unsatisfiability of $\mathrm{prop}(C)$. This means that $\texttt{GetSATModel}(\mathrm{prop}(C), \emptyset) = \texttt{NIL}$. Therefore, $\mathcal{ALCCQU}$-BB(C) returns **false**.

Termination is guaranteed, because GetSATModel is assumed to implement a correct and terminating decision procedure for propositional satisfiability. □

We proceed by showing that the subprocedure of $\mathcal{ALCCQU}$-BB that employs branch-and-price to solve a CQU instance induced from a set of role successor constraints — lines 8–20 — is correct and terminates. The proof relies on the correctness and termination of the branch-and-bound and column generation methods (details in [10, 19]).

In particular, column generation terminates and is correct because the matrix A is finite and for every added column the objective function is not increased — the only additional factor we need to take care of is to avoid adding columns that are already in the restricted master problem (4.8) [10]. The termination of branch-and-bound is guaranteed when the solution space is bounded, that is,

when there exist a vector $\mathbf{t}$ such that $\mathbf{0} \leq \mathbf{x} \leq \mathbf{t}$. In our setting, the upper bound $\mathbf{t}$ is not guaranteed to exist *a priori*; however, linear programming ensures that if a solution to the problem (4.7) exists, then there exists a solution satisfying $x_j \leq \max\{b_i \mid \mathbf{b} = (b_1, \ldots, b_k)\}$, thus we can impose an artificial upper bound and ensure the termination of the branch and bound [11, 19].

The section of $\mathcal{ALCCQU}$-BB ranging over lines 8–20 is entered only if $\mathrm{prop}(C)$ has a model μ that has not being tested yet and if $\texttt{GetConstraints}(C, \mu) \neq \emptyset$. Therefore, we can assume that $\texttt{cd}(C) = n + 1$, with $n \geq 0$ a natural number.

Theorem 19. *The $\mathcal{ALCCQU}$ concept C in normal form is satisfiable iff $\mathcal{ALCCQU}$-BB(C) returns **true**. Moreover, $\mathcal{ALCCQU}$-BB terminates.*

Proof. By induction on $\texttt{cd}(C)$. The induction base $\texttt{cd}(C) = 0$ is covered in Lemma 10. We assume the following inductive hypothesis, for every $\mathcal{ALCCQU}$ concept description D:

$$\text{if } \texttt{cd}(D) \leq n, \text{ then } D \text{ is satisfiable if and only if } \mathcal{ALCCQU}\text{-BB}(D) = \textbf{true}.$$

Assume that $\texttt{cd}(C) = n + 1$.

Soundness Assume that $\mathcal{ALCCQU}$-BB$(C) =$ **true**. Then, there exists a truth assignment μ such that $\mu \models \mathrm{prop}(C)$. If $\texttt{GetConstraints}(C, \mu) = \emptyset$, then the interpretation $\mathcal{I}$ with $d_0 \in \Delta^{\mathcal{I}}$ defined by

$$d_0 \in A^{\mathcal{I}} \text{ if and only if } \mu(x_A) = 1 \text{ for all } \mathcal{ALCCQU}\text{-atoms } A \text{ in } C$$

is a model for C. Assume now that $\texttt{GetConstraints}(C, \mu) \neq \emptyset$ and let $\psi = \langle \mathcal{Q}, \mathcal{U} \rangle$ be the CQU formula in normal form obtained from $\texttt{GetConstraints}(C, \mu)$. Since we assumed that $\mathcal{ALCCQU}$-BB(C) returns **true**, there is an integral solution $\mathbf{x} = (x_1, \ldots, x_m)$ to the master problem (4.7) associated to ψ. Let $\mathbf{v} = A_j$ be a column of the matrix A (4.7) such that $x_j \neq 0$. Then, $\mathbf{v}$ is $\mathcal{U}$-satisfying and we can assume that there exists a valuation μ_v that satisfies $\mathrm{prop}(\mathcal{U})$ and yields $\mathbf{v}$. Thanks to our inductive hypothesis, since $\texttt{cd}(C_v) \leq n$, we also know that the concept C_v yielded by $\mathbf{v}$ and μ is satisfiable: thus,we can consider a model $\mathcal{I}_v$ of C_v and an individual $d_v \in \Delta^{\mathcal{I}_v}$ such that $d_v \in C_v^{\mathcal{I}_v}$. We take x_j disjoint copies of $\mathcal{I}_v$ and embed them in the interpretation $\mathcal{I}$ by connecting each individual d_v to d_0 as follows:

$$d_v \in r^{\mathcal{I}}(d_0) \text{ if and only if } \mu_v(R_r) = 1.$$

Since $\mu_v \models \bigvee_{r \in N_R} R_r$, there exists at least a role name r such that $d_v \in r^{\mathcal{I}}(d_0)$, thus d_v is a proper role successor. Moreover, if $|\mathcal{Q}| = k$, for $i = 1, \ldots, k$ we obtain that

$$d_v \in s_i^{\mathcal{I}_{d_0}} \text{ if and only if } v_i = 1$$

where s_i is the set term referenced by the i-th counting sentence in $\mathcal{Q}$. We repeat this procedure for all the columns A_j such that $x_j \neq 0$.

For $i = 1, \ldots, k$, let $S_i := \{x_j \mid \mathbf{v} = A_j \wedge v_i = 1\}$. For every counting sequence $\exists_{\bowtie_i N_i} x.p_i(x)$ in $\mathcal{Q}$ representing the successor constraint $\mathrm{succ}(|s_i| \bowtie_i N_i)$, we obtain that

$$\sum_{j=1}^{m} A_{j_i} x_j = \left(\sum_{\substack{x_j \in S_i \\ x_j \neq 0}} x_j \right) \bowtie_i N_i$$

thus $d_0 \in \mathrm{succ}(|s_i| \bowtie_i N_i)^{\mathcal{I}}$ for $i = 1, \ldots, k$. We conclude that $\mathcal{I} \models C$.

Completeness assume that C is satisfiable under the interpretation $\mathcal{I}$ and that $d_0 \in C^{\mathcal{I}}$. Let μ be the truth assignment defined by

$$\mu(x_A) = 1 \text{ if and only if } d_0 \in A^{\mathcal{I}} \text{ for all } \mathcal{ALCCQU}\text{-atoms } A \text{ in } C.$$

Then, $\mu \models \mathrm{prop}(C)$; since $noGoods = \emptyset$, the algorithm $\mathcal{ALCCQU}$-BB proceeds. If $S := \texttt{GetConstraints}(C, \mu) = \emptyset$, $\mathcal{ALCCQU}$-BB returns **true**. Assume that $S \neq \emptyset$. The CQU formula in normal form ψ obtained from S is satisfiable, thanks to Lemma 8 and Theorem 18. Moreover, from Theorem 16 follows that the master problem (4.7) has a feasible and integral solution. Since the branch-and-price method is proved to be correct, an integral solution is found for one of the subproblems generated from ψ, thus $\mathcal{ALCCQU}$-BB returns **true**.

Termination Both the branch-and-bound and the column generation methods are guaranteed to terminate [10, 19]. Moreover, every recursive call to $\mathcal{ALCCQU}$-BB comes with a constraint depth that is at most n; hence, by inductive hypothesis, the recursive call terminates. Finally, the propositional formula $\mathrm{prop}(C)$ has a finite number of truth assignments that are models.

□

4.4.1 Complexity of $\mathcal{ALCCQU}$-BB

A property that is enjoyed by the original decision procedure for $\mathcal{ALCSCC}$ concept satisfiability in [2] is that it can be adapted to use only a polynomial amount of space with respect to the input concept description. Since we opted to replace non-determinism with other techniques, our newly-defined decision procedure does not enjoy this property. In particular:

- We store all the models of $\mathrm{prop}(C)$ that yield a failing run of the algorithm; in the worst case, $\mathrm{prop}(C)$ might have exponentially many models to test, in the size of the set of its $\mathcal{ALCCQU}$-atoms.
- Every time we check if a CQU formula is satisfiable in $\mathcal{ALCCQU}$-BB, we resort to column generation. Since we are not guessing a polynomial-sized set of columns as done in [2] but we are generating as many columns as needed in order to find a satisfying solution, in the worst case an exponential number of columns in the size of the number of counting sentences in the CQU formula. Moreover, every call to the SAT oracle to obtain a column invokes a NP-procedure.

- The algorithm $\mathcal{ALCCQU}$-BB requires that the input concept description C is in normal form. This means that the normalization preprocessing might already take exponential time and yield an exponentially larger concept description $\mathtt{nf}(C)$.

During the column generation procedure, we have a worst-case exponential number of calls to a NP subprocedure. This implies that $\mathcal{ALCCQU}$-BB is a NExpTime-algorithm to decide $\mathcal{ALCCQU}$ concept satisfiability. We remark the fact that this is a worst-case complexity result and that in practice, the execution of $\mathcal{ALCCQU}$-BB might be more efficient. The only way to find an answer to this question is to provide an implementation of $\mathcal{ALCCQU}$-BB and evaluate its performance.

5 Conclusion

Summary. Starting from the variant $\mathcal{ALCSCC}^{\infty}$ of an existing extension of $\mathcal{ALCQ}$ called $\mathcal{ALCSCC}$ [2] with known reasoning complexity and decision procedures for concept satisfiability, we proposed an algorithm that can be efficiently implemented to test concept satisfiability in a restriction of $\mathcal{ALCSCC}^{\infty}$ called $\mathcal{ALCCQU}$, where the only cardinality constraints that are allowed are comparisons of complex cardinality terms against a constant natural number. The algorithm that we presented relies on a variety of known techniques borrowed from SAT solving and integer linear programming, applied to a first-order fragment called CQU [11] that we adapted to represent the constraints over role successors of a given individual. This decision procedure successfully replaces the non-deterministic steps of the original method to check $\mathcal{ALCSCC}$ concept satisfiability proposed in [2], at the cost of losing some complexity-related properties regarding space requirements.

We have also shown that $\mathcal{ALCCQU}$ corresponds to the subset of $\mathcal{ALCSCC}^{\infty}$ that lies in first-order logic. This property strengthens our choice of analyzing $\mathcal{ALCCQU}$ on the one hand and shows that the technique illustrated here cannot be applied to the whole DL $\mathcal{ALCSCC}^{\infty}$ without restrictions on the other, since there exist $\mathcal{ALCSCC}^{\infty}$ concepts that are beyond first-order logic. To provide an additional argument in favor of studying $\mathcal{ALCCQU}$, we have classified the DLs $\mathcal{ALCQ}$, $\mathcal{ALCCQU}$ and $\mathcal{ALCSCC}^{\infty}$ according to their expressive power, obtaining a linear order

$$\mathcal{ALCQ} <_{\text{expr}} \mathcal{ALCCQU} <_{\text{expr}} \mathcal{ALCSCC}^{\infty}$$

that shows how $\mathcal{ALCCQU}$ constitutes a middle ground between the well-known DL $\mathcal{ALCQ}$ and the rather expressive DL $\mathcal{ALCSCC}^{\infty}$. Moreover, we presented a third characterization of $\mathcal{ALCCQU}$ as the first-order fragment that is invariant under the newly-introduced equivalence relation of $\mathcal{ALCQ}t$-bisimulation.

Future work. To substantiate the claim that the algorithm proposed in Chapter 4 is efficient for practical purposes, the different decision procedures for $\mathcal{ALCSCC}$ and $\mathcal{ALCCQU}$ ought to be implemented and evaluated on a subset of instances that can be expressed by both logics. Using the implementation of CQU-SAT provided in [11] could be a starting point for the implementation of $\mathcal{ALCCQU}$-BB.

Another interesting task would be to investigate finite-model reasoning over $\mathcal{ALCSCC}^{\infty}$ and $\mathcal{ALCCQU}$. In such a setting, indeed, the set of role successors of an individual is always bounded, leading to additional guarantees — for instance, Lemma 1 always holds in a finite model. Alternatively, one could investigate the consequences of setting an upper bound to the set of role successors: in that case, if the bound is known, all possible role successor constraints of $\mathcal{ALCSCC}^{\infty}$ could be translated into first-order logic, due to the finiteness of the numerical domains that are considered.

The algorithm that we proposed can be extended in a straightforward fashion to acyclic TBoxes, by unfolding the definitions in the input concept description C. A more interesting question could be whether we can take a similar approach to reason with respect to general TBoxes, possibly by employing suitable blocking strategies.

Finally, it would be worth considering the interaction of $\mathcal{ALCCQU}$ and $\mathcal{ALCSCC}^{\infty}$ with CBoxes (or their restricted versions) as in [3] and understand if our approach based on linear programming could be employed in such a setting. A positive outcome would yield an efficient tool to provide statistical reasoning in a DL, as described in [3].

Bibliography

[1] G. E. Andrews. *The theory of partitions*. 1st paperback ed. Cambridge mathematical library. Cambridge: Cambridge Univ. Press, 1998. 255 pp. ISBN: 978-0-521-63766-4.

[2] F. Baader. "A New Description Logic with Set Constraints and Cardinality Constraints on Role Successors". In: *Frontiers of Combining Systems*. Ed. by C. Dixon and M. Finger. Vol. 10483. Springer International Publishing, 2017, pp. 43–59. ISBN: 978-3-319-66166-7 978-3-319-66167-4. DOI: 10.1007/978-3-319-66167-4_3.

[3] F. Baader and A. Ecke. "Extending the Description Logic ALC with More Expressive Cardinality Constraints on Concepts". In: GCAI 2017. 3rd Global Conference on Artificial Intelligence. Red. by C. Benzmüller, C. Lisetti, and M. Theobald. Vol. 50. EPiC Series in Computing. 2017, pp. 6–19. DOI: 10.29007/f3hh.

[4] F. Baader, I. Horrocks, C. Lutz, and U. Sattler. *An Introduction to Description Logic*. Cambridge University Press, 2017. ISBN: 978-0-521-69542-8.

[5] F. Baader, E. Karabaev, C. Lutz, and M. Theißen. "A New n-Ary Existential Quantifier in Description Logics". In: *KI 2005: Advances in Artificial Intelligence*. Ed. by U. Furbach. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 18–33. ISBN: 978-3-540-31818-7.

[6] C. Barnhart, E. L. Johnson, G. L. Nemhauser, M. W. P. Savelsbergh, and P. H. Vance. "Branch-and-Price: Column Generation for Solving Huge Integer Programs". In: *Operations Research* 46.3 (June 1998), pp. 316–329. ISSN: 0030-364X, 1526-5463. DOI: 10.1287/opre.46.3.316.

[7] D. Bertsimas and J. Tsitsiklis. *Introduction to Linear Optimization*. Jan. 1998.

[8] C. C. Chang and H. J. Keisler. *Model theory*. 3rd ed. Studies in logic and the foundations of mathematics v. 73. Amsterdam ; New York : New York, NY, USA, 1990. 650 pp. ISBN: 978-0-444-88054-3.

[9] D. van Dalen. *Logic and structure (3. ed.)* Universitext. Springer, 1994. ISBN: 978-3-540-57839-0.

[10] J. Desrosiers and M. E. Lübbecke. "A Primer in Column Generation". In: *Column Generation*. Ed. by G. Desaulniers, J. Desrosiers, and M. M. Solomon. Boston, MA: Springer US, 2005, pp. 1–32. ISBN: 978-0-387-25486-9. DOI: 10.1007/0-387-25486-2_1.

[11] M. Finger and G. D. Bona. "Algorithms for Deciding Counting Quantifiers over Unary Predicates". In: *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*. 2017, pp. 3878–3884.

[12] V. Haarslev, R. Sebastiani, and M. Vescovi. "Automated Reasoning in $\mathcal{ALCQ}$ via SMT". In: *Automated Deduction – CADE-23*. Ed. by N. Bjørner and V. Sofronie-Stokkermans. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2011, pp. 283–298. ISBN: 978-3-642-22438-6.

[13] P. Hall. "On Representatives of Subsets". In: *Journal of the London Mathematical Society* s1-10.1 (1935), pp. 26–30. DOI: 10.1112/jlms/s1-10.37.26.

[14] N. Z. Karahroodi and V. Haarslev. "A Consequence-based Algebraic Calculus for SHOQ". In: *Proceedings of the 30th International Workshop on Description Logics, Montpellier, France, July 18-21, 2017*. Ed. by A. Artale, B. Glimm, and R. Kontchakov. Vol. 1879. CEUR Workshop Proceedings. CEUR-WS.org, 2017.

[15] V. Kuncak and M. Rinard. "Towards Efficient Satisfiability Checking for Boolean Algebra with Presburger Arithmetic". In: *Automated Deduction – CADE-21*. Ed. by F. Pfenning. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2007, pp. 215–230. ISBN: 978-3-540-73595-3.

[16] C. Lutz, R. Piro, and F. Wolter. "Description Logic TBoxes: Model-Theoretic Characterizations and Rewritability". In: *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011*. 2011, pp. 983–988. DOI: 10.5591/978-1-57735-516-8/IJCAI11-169.

[17] M. E. Lübbecke and J. Desrosiers. "Selected Topics in Column Generation". In: *Operations Research* 53.6 (Dec. 2005), pp. 1007–1023. ISSN: 0030-364X, 1526-5463. DOI: 10.1287/opre.1050.0234.

[18] A. Rodríguez-González, J. Torres-Niño, G. Hernández-Chan, E. Jiménez-Domingo, and J. M. Alvarez-Rodríguez. "Using agents to parallelize a medical reasoning system based on ontologies and description logics as an application case". In: *Expert Systems with Applications* 39.18 (2012), pp. 13085–13092. ISSN: 0957-4174. DOI: 10.1016/j.eswa.2012.05.093.

[19] A. Schrijver. "Branch-and-Bound Methods for Integer Linear Programming". In: *Theory of Linear and Integer Programming*. New York, NY, USA: John Wiley & Sons, Inc., 1986, pp. 360–362. ISBN: 0-471-90854-1.

[20] S. Tobies. "Complexity Results and Practical Algorithms for Logics in Knowledge Representation". PhD thesis, pp. 49–51.

[21] J. P. Warners. "A Linear-Time Transformation of Linear Inequalities into Conjunctive Normal Form". In: *Inf. Process. Lett.* 68.2 (1998), pp. 63–69. DOI: 10.1016/S0020-0190(98)00144-6.