On the Expressive Power of Description Logics with Cardinality Constraints on Finite and Infinite Sets[⋆]

Franz Baader and Filippo De Bortoli

Theoretical Computer Science, TU Dresden, Dresden, Germany
franz.baader@tu-dresden.de, filippo.de_bortoli@tu-dresden.de

Abstract. In recent work we have extended the description logic (DL) $\mathcal{ALCQ}$ by means of more expressive number restrictions using numerical and set constraints stated in the quantifier-free fragment of Boolean Algebra with Presburger Arithmetic (QFBAPA). It has been shown that reasoning in the resulting DL, called $\mathcal{ALCSCC}$, is PSpace-complete without a TBox and ExpTime-complete w.r.t. a general TBox. The semantics of $\mathcal{ALCSCC}$ is defined in terms of finitely branching interpretations, that is, interpretations where every element has only finitely many role successors. This condition was needed since QFBAPA considers only finite sets. In this paper, we first introduce a variant of $\mathcal{ALCSCC}$, called $\mathcal{ALCSCC}^\infty$, in which we lift this requirement (inexpressible in first-order logic) and show that the complexity results for $\mathcal{ALCSCC}$ mentioned above are preserved. Nevertheless, like $\mathcal{ALCSCC}$, $\mathcal{ALCSCC}^\infty$ is not a fragment of first-order logic. The main contribution of this paper is to give a characterization of the first-order fragment of $\mathcal{ALCSCC}^\infty$. The most important tool used in the proof of this result is a notion of bisimulation that characterizes this fragment.

1 Introduction

Description Logics (DLs) [4] are a well-investigated family of logic-based knowledge representation languages, which are frequently used to formalize ontologies for application domains such as biology and medicine [8]. To define the important notions of such an application domain as formal concepts, DLs state necessary and sufficient conditions for an individual to belong to a concept. These conditions can be Boolean combinations of atomic properties required for the individual (expressed by concept names) or properties that refer to relationships with other individuals and their properties (expressed as role restrictions). For example, the concept of a man that has a son and a daughter can be formalized in the DL $\mathcal{ALC}$ [17] as $\mathit{Male} \sqcap \mathit{Human} \sqcap \exists \mathit{child}.\mathit{Male} \sqcap \exists \mathit{child}.\mathit{Female}$. Number restrictions allow us to formulate numerical constraints on the role successors of the elements of a concept. For example, in the DL $\mathcal{ALCQ}$ [9], the concept

[⋆] Partially supported by the German Research Foundation (DFG) within the Research Unit 1513 (Hybris) and the Research Training Group 1763 (QuantLA).

$\mathit{Female} \sqcap \mathit{Human} \sqcap (\geq 5\, \mathit{child} \,.\, \mathit{Female}) \sqcap (\leq 1\, \mathit{child} \,.\, \mathit{Male})$ describes women that have at least five daughters and at most one son.

In recent work [2], we have extended the DL $\mathcal{ALCQ}$ by means of more expressive number restrictions using numerical and set constraints stated in the quantifier-free fragment of Boolean Algebra with Presburger Arithmetic (QFBAPA) [11]. For example, in the resulting DL $\mathcal{ALCSCC}$, one can describe individuals that have twice as many sons as daughter using the role successor constraint $\mathit{succ}(|\mathit{child} \cap \mathit{Male}| = 2 \cdot |\mathit{child} \cap \mathit{Female}|)$. It has been shown in [2] that reasoning in $\mathcal{ALCSCC}$ has the same complexity as reasoning in its sub-logic $\mathcal{ALCQ}$ [18,19], i.e., PSpace-complete without a TBox and ExpTime-complete in the presence of a TBox.

The semantics of $\mathcal{ALCSCC}$ is defined in terms of finitely branching interpretations, i.e., interpretations where every element has only finitely many role successors. This condition was needed since QFBAPA considers only finite sets. The disadvantage of this meta-condition is that it is not expressible in first-order logic, and thus makes the comparison of the expressive power of $\mathcal{ALCSCC}$ with that of other DLs, which are usually fragments of first-order logic, problematic. Strictly speaking, no $\mathcal{ALCSCC}$ concept is expressible in first-order logic due to this implicit constraint. To overcome this problem, we introduce a variant of $\mathcal{ALCSCC}$, called $\mathcal{ALCSCC}^\infty$, in which we lift the "finite branching" requirement. This is achieved by introducing a variant of QFBAPA, called QFBAPA$^\infty$, in which not just finite, but also infinite sets are considered. We prove that satisfiability in QFBAPA$^\infty$ has the same complexity (NP-complete) as satisfiability in QFBAPA. Based on this, we can show that the complexity results for $\mathcal{ALCSCC}$ mentioned above also hold for $\mathcal{ALCSCC}^\infty$. Alternatively, we could have also used the variant QFBAPA$_\infty$ of QFBAPA with possibly infinite sets introduced in [10], whose satisfiability problem is also NP-complete.

Despite the removal of the "finite branching" requirement — a constraint that destroys expressibility in first-order logic of $\mathcal{ALCSCC}$— $\mathcal{ALCSCC}^\infty$ is still not a fragment of first-order logic. The main contribution of this paper is to give a characterization of the first-order fragment of $\mathcal{ALCSCC}^\infty$. For this purpose, we introduce the fragment $\mathcal{ALCCQU}$ of $\mathcal{ALCSCC}^\infty$, which uses constraints of the logic CQU (counting quantifiers over unary predicates) [7] in place of QFBAPA constraints, and show that $\mathcal{ALCCQU}$ concepts are first-order definable. Basically, in $\mathcal{ALCCQU}$ we can compare the cardinality of successors sets with a constant, but not with the cardinality of another successor set. For example, the $\mathcal{ALCCQU}$ role successor constraint $\mathit{succ}(|\mathit{friend} \cap \mathit{livesWith} \cap \mathit{Female}| \geq 2)$ describes individuals that live together with at least two female friends. To get a handle on the expressive power of $\mathcal{ALCCQU}$, we define a notion of bisimulation that characterizes $\mathcal{ALCCQU}$, in the sense that a first-order formula is invariant under this kind of bisimulation iff it is equivalent to an $\mathcal{ALCCQU}$ concept. This notion of bisimulation is very similar to the counting bisimulation introduced in [13] for $\mathcal{ALCQ}$. Surprisingly, all $\mathcal{ALCSCC}^\infty$ concepts are also invariant under $\mathcal{ALCCQU}$-bisimulation, which allows us to conclude that $\mathcal{ALCCQU}$ consists of exactly the first-order definable concepts of $\mathcal{ALCSCC}^\infty$.

When formulating complexity results for logics that use numbers in their syntax (such as QFBAPA, CQU, $\mathcal{ALCCQU}$, and $\mathcal{ALCSCC}^\infty$), it is important to make clear how the input size of the numbers is defined. In this paper, we assume *binary coding* of numbers (e.g., the number 1024 has size 10 rather than size 1024), which makes the complexity upper bounds stronger.

2 The logics QFBAPA$^\infty$ and CQU

In this section, we first introduce the logic QFBAPA$^\infty$, whose main difference to the well-known logic QFBAPA [11] is that it allows for solutions involving not just finite, but also infinite sets. Then, we demonstrate that two important results shown in [11] for QFBAPA also hold for QFBAPA$^\infty$. Finally, we define the fragment CQU of QFBAPA$^\infty$, for which a decision procedure based on column generation was described in [7].

The logic QFBAPA$^\infty$ In this logic one can build *set terms* by applying Boolean operations (intersection $\cap$, union $\cup$, and complement $\cdot^c$) to set variables as well as the constants $\emptyset$ and $\mathcal{U}$. Set terms s, t can then be used to state inclusion and equality constraints ($s = t, s \subseteq t$) between sets. *Presburger Arithmetic (PA) expressions* are built from non-negative integer constants, PA variables, and set cardinalities $|s|$ using addition as well as multiplication with a non-negative integer constant. They can be used to form numerical constraints of the form $k = \ell$ and $k < \ell$, where k, ℓ are PA expressions. A *QFBAPA$^\infty$ formula* is a Boolean combination of set and numerical constraints.

The semantics of set terms and set constraints is defined using *substitutions* σ that assign a set $\sigma(\mathcal{U})$ to $\mathcal{U}$ and subsets of $\sigma(\mathcal{U})$ to set variables. The evaluation of set terms and set constraints by such a substitution is defined in the obvious way, using the standard notions of intersection, union, complement,[1] inclusion, and equality for sets. PA expressions are evaluated over $\mathbb{N}^\infty = \mathbb{N} \cup \{\infty\}$, i.e., the non-negative integers extended with a symbol for infinity. Thus, substitutions additionally assign elements of $\mathbb{N}^\infty$ to PA variables. The cardinality expression $|s|$ is evaluated under σ as the cardinality of $\sigma(s)$ if this set is finite, and as ∞ if $\sigma(s)$ is not finite.[2] When evaluating PA expressions w.r.t. a substitution σ, we employ the usual way of adding, multiplying, and comparing integers, extended to $\mathbb{N}^\infty$ by the following rules ranging over $N \in \mathbb{N}$:

1. $\infty + N = N + \infty = \infty = \infty + \infty$,
2. if $N \neq 0$ then $N \cdot \infty = \infty = \infty \cdot N$, else $0 \cdot \infty = 0 = \infty \cdot 0$,
3. $N < \infty$ and $\infty \not< N$, as well as $\infty = \infty$ and $\infty \not< \infty$.

A *solution* σ of a QFBAPA$^\infty$ formula ϕ is a substitution that evaluates ϕ to true, using the above rules for evaluating set and numerical constraints and the

[1] The complement is defined w.r.t. $\sigma(\mathcal{U})$, i.e., $\sigma(s^c) = \sigma(\mathcal{U}) \setminus \sigma(s)$.

[2] Note that we do not distinguish between different infinite cardinalities, such as countably infinite, uncountably infinite, etc.

usual interpretation of the Boolean operators occurring in ϕ. The formula ϕ is *satisfiable* if it has a solution.

Note that, in QFBAPA$^\infty$, we can enforce infinity of a set although we do not allow the use of ∞ as a constant. For instance, $|s| = \infty$ is not an admissible numerical constraint, but it is easy to see that the constraint $|s| + 1 = |s|$ can only be satisfied by a substitution that assigns an infinite set to the set term s.

Comparison with QFBAPA and QFBAPA$_\infty$ The logic QFBAPA, as introduced in [11], differs from QFBAPA$^\infty$ as defined above, both syntactically and semantically. From the syntactic point of view, the main difference is that, in QFBAPA$^\infty$, we disallow negative integer constants as well as divisibility by a fixed integer constant. Dispensing with negative constants is not really a restriction since we can always write the numerical constraints of QFBAPA in a way that does not use negative integer constants (by bringing negative summands to the other side of a constraint). Disallowing divisibility may be a real restriction, but in the presence of ∞ it is not clear how to interpret divisibility constraints (Is ∞ even or odd?). In addition, in the context of using the logic within a DL, there does not appear to be an urgent need for such constraints. From a syntactic point of view, the logic QFBAPA$_\infty$ [10] has more general atomic constraints than our logic QFBAPA$^\infty$ (e.g., it allows for the use of rational constants and the explicit statement that a set is infinite), but these constraints can be expressed also in QFBAPA$^\infty$.

From the semantic point of view, the main difference of QFBAPA$^\infty$ and QFBAPA$_\infty$ to QFBAPA is, of course, that the semantics of QFBAPA requires us to interpret $\mathcal{U}$, and thus all set variables, as finite sets. In addition, PA variables are interpreted in QFBAPA$^\infty$ as non-negative integers, but this was already the case for the variant of QFBAPA used in the definition of $\mathcal{ALCSCC}$ in [2] since in that context only set cardinalities (which are non-negative) are used. In QFBAPA$_\infty$, PA variables can also be interpreted as real numbers, but there is a constraint available that allow to state that a PA term must be interpreted by an integer. Since PA variables are not used in the context of $\mathcal{ALCSCC}$ and $\mathcal{ALCSCC}^\infty$, this difference is not relevant here.

Satisfiability in QFBAPA, QFBAPA$^\infty$, and QFBAPA$_\infty$ In [11] it is shown that the satisfiability problem for QFBAPA formulae is NP-complete. Since NP-hardness is clear due to the use of Boolean operations on the formula level, the main task in [11] was to show the "in NP" result. The main tool used in [11] is a "sparse solution" lemma (see Fact 1 in [11] and Lemma 3 in [2]), which was also important for showing the complexity upper bounds for reasoning in $\mathcal{ALCSCC}$ in [2]. We show below that this "sparse solution" lemma also holds for QFBAPA$^\infty$, which implies that satisfiability of QFBAPA$^\infty$ formulae is also in NP.

The "sparse solution" lemma is based on the notion of a Venn region. Assume that ϕ is a QFBAPA formula containing the set variables $X_1, \ldots, X_k$. A *Venn* *region* for ϕ is of the form

$$X_1^{c_1} \cap \ldots \cap X_k^{c_k},$$

where c_i is either empty or c for $i = 1, \ldots, k$. Venn regions are interesting since every set term s occurring in ϕ can be expressed as the disjoint union of Venn regions, and thus its cardinality is the sum of the cardinalities of these Venn regions. The problem is that there may be exponentially many Venn regions in the size of ϕ. The "sparse solution" lemma basically says that it is possible to restrict the attention to a polynomial number of Venn regions.

To be more precise, it is shown in [11] that a given QFBAPA formula ϕ can be transformed in polynomial time into an equisatisfiable QFBAPA formula $G \wedge F$ where G is a Boolean combination of numerical constraints not containing sets, and F is a conjunction of linearly many expressions $|b_i| = k_i$ ($i = 1, \ldots, m$), where b_i is a set term and k_i is a PA variable (standing for the cardinality of the set b_i). Using the fact that each b_i is a disjoint union of Venn regions, F can be expressed as a system of linear equations

$$A \cdot x = k, \tag{1}$$

which must be solved over the non-negative integers. Here the ith row of the matrix A is a 0/1 vector that expresses which Venn regions participate in generating the set b_i. The vector x contains a variable x_v for every Venn region v at the position corresponding to the occurrence of this region in A. Intuitively, the value of x_v in a solution of the equation stands for the cardinality of the Venn region. Finally, k is the vector of the variables k_i, and thus the value of k_i stands for the cardinality of the set b_i.

For example, consider the QFBAPA formula

$$\phi = |X_1 \cup X_2| \geq |X_1| + |X_2|.$$

In this case, G is $k_1 \geq k_2 + k_3$ and F is $|X_1 \cup X_2| = k_1 \wedge |X_1| = k_2 \wedge |X_2| = k_3$, and thus $m = 3$ is the number of rows in the matrix A. There are four Venn regions in this example: $v_1 = X_1 \cap X_2$, $v_2 = X_1 \cap X_2^c$, $v_3 = X_1^c \cap X_2$, $v_4 = X_1^c \cap X_2^c$. The system (1) thus has four variables in the vector x, and looks as follows:

$$\begin{pmatrix} 1\,1\,1\,0 \\ 1\,1\,0\,0 \\ 1\,0\,1\,0 \end{pmatrix} \cdot \begin{pmatrix} x_{v_1} \\ x_{v_2} \\ x_{v_3} \\ x_{v_4} \end{pmatrix} = \begin{pmatrix} k_1 \\ k_2 \\ k_3 \end{pmatrix} \tag{2}$$

The first row of A in (2) is explained by the fact that $X_1 \cup X_2 = v_1 \cup v_2 \cup v_3$, and similarly for the other rows. If we take the solution $k_1 = 3, k_2 = 1, k_3 = 1$ of G, then the linear system obtained from (2) by applying this replacement is actually not solvable. In contrast, if we take $k_1 = 2, k_2 = 1, k_3 = 1$, then setting $x_{v_1} = 0$, $x_{v_2} = 1$, $x_{v_3} = 1$, $x_{v_4} = 5$ is a solution of the resulting system (where the value for x_{v_4} is actually irrelevant). Note that $x_{v_1} = 0$ means that the Venn region $v_1 = X_1 \cap X_2$ is empty, which corresponds to the fact that we can only have $|X_1 \cup X_2| \geq |X_1| + |X_2|$ if X_1 and X_2 are interpreted by disjoint sets.

The problem with the system (1) is that it contains exponentially many variables. Using a result by Eisenbrand and Shmonin [6] (called Fact 1 in [11]), it is shown in [11] that there is a bound $N = 2m \log(4m)$ such that, for any solution of G, the system obtained from (1) by applying this solution to the variables k_i has a solution in which at most N of the variables x_v in (1) are non-zero, if it has a solution at all. Note that the value of N is clearly polynomial in the size of ϕ since m is bounded by the number of set terms occurring in ϕ.

On the one hand, this implies that, by guessing (in non-deterministic polynomial time) N Venn regions v whose associated variables in (1) are supposed to be non-zero, we obtain a polynomial-sized system $A' \cdot x' = k$, in which only the variables x_v for the guessed Venn regions and their associated columns remain. The formula G is a Boolean combination of linear (in)equations. After guessing which of them are to be true and which not, we overall obtain a polynomially large system of linear (in)equations, whose solvability in the non-negative integers $\mathbb{N}$ can then be tested by an NP procedure [14]. Note that this NP procedure is not used as an NP oracle, but rather as an extension of the search tree for a solution that has already been built by the guessing done before.

Proposition 1 ([11]). *Satisfiability of QFBAPA formulae is in NP.*

On the other hand, since $x_v = 0$ means that the Venn region v is empty, the above argument also shows that a solvable QFBAPA formula always has a solution in which at most N Venn regions are non-empty. In [2] this result was actually strengthened as follows.

Lemma 1 ([2]). *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 the following holds for every solution σ of ϕ: there is a solution σ' of ϕ such that*

(i) $|\{v \text{ Venn region } \mid \sigma'(v) \neq \emptyset\}| \leq N$, *and*
(ii) $\{v \text{ Venn region } \mid \sigma'(v) \neq \emptyset\} \subseteq \{v \text{ Venn region } \mid \sigma(v) \neq \emptyset\}$.

In the corresponding result shown in [11], the property (ii) is missing. The main idea underlying the proof of our stronger result is that, for a given solution σ of ϕ, one applies the "sparse solution" lemma of [6] not to (1) directly, but to the system obtained from it by removing the variables x_v and the corresponding columns in A for those Venn regions v that satisfy $\sigma(v) = \emptyset$.

Our goal is now to show that Proposition 1 and Lemma 1 also hold if we replace QFBAPA with QFBAPA$^\infty$. For this purpose, let us assume that σ is a solution of G in $\mathbb{N}^\infty$, and consider the system

$$A \cdot x = \sigma(k), \tag{3}$$

where the variables k_i are replaced by $\sigma(k_i) \in \mathbb{N}^\infty$. Then the following lemma is an easy consequence of the way we defined operations involving ∞.

Lemma 2. *The following holds for any solution θ of (3): if $\sigma(k_i) = \infty$, then there is a Venn region v such that $\theta(x_v) = \infty$ and*

1. *the column in A corresponding to v contains 1 at position position i, and*
2. *for all j with $\sigma(k_j) < \infty$, the column in A corresponding to v contains 0 at position j.*

Based on this lemma, we can now extend the "sparse solution" lemma of [6] to the setting where ∞ may occur on the right-hand side of the system and in the solution.

Lemma 3. *If the linear system (3) has a solution in $\mathbb{N}^\infty$, then it has a solution in $\mathbb{N}^\infty$ where at most $M = 2m\log(4m) + m$ of the variables in the vector x are non-zero.*

Proof. Assume that (3) has a solution θ in $\mathbb{N}^\infty$. Then the system $B \cdot x = b$ obtained from (3) by removing the rows i for which $\sigma(k_i) = \infty$ also has a solution in $\mathbb{N}^\infty$. In addition, since the vector b does not contain ∞, it is easy to see that $B \cdot x = b$ also has a solution in $\mathbb{N}$. The "sparse solution" lemma of [6] then yields a solution γ of $B \cdot x = b$ such that at most $N = 2m\log(4m)$ of the variables x_v in x are such that $\gamma(x_v) \neq 0$.

We modify γ to obtain a solution γ' of (3) using Lemma 2. Since (3) has the solution θ, we know the following: for every i such that $\sigma(k_i) = \infty$, there is a Venn region v such that 1. and 2. stated in that lemma are satisfied. We now select for each such i one Venn region v with these properties, and set $\gamma'(x_v) := \infty$. For Venn regions u not selected in this way, we set $\gamma'(x_u) := \gamma(x_u)$. In the worst-case, this modification changes m zeros to ∞, and thus γ' satisfies the bound M on the number of non-zero variables. It remains to show that γ' solves (3). Thus, consider the ith equation in this system. If $\sigma(k_i) = \infty$, then there is a v such that $\gamma'(x_v) = \infty$ and 1. in Lemma 2 is satisfied. This implies that γ' solves the ith equation. If $\sigma(k_i) = b_i < \infty$, then 2. in Lemma 2 implies that the modifications made to obtain γ' from γ have no effect on this equation since the modified values are multiplied with 0, and thus removed. Hence, we have shown that γ' solves (3) and it satisfies the required bound on the number of non-zero variables. □

This lemma allows us to extend Proposition 1 and Lemma 1 to QFBAPA$^\infty$.

Theorem 1. *Satisfiability of QFBAPA$^\infty$ formulae is in NP. Moreover, for every QFBAPA$^\infty$ formula ϕ, one can compute in polynomial time a number N whose value is polynomial in the size of ϕ such that the following holds for every solution σ of ϕ: there is a solution σ' of ϕ such that*

(i) $|\{v \text{ Venn region } \mid \sigma'(v) \neq \emptyset\}| \leq N$, *and*
(ii) $\{v \text{ Venn region } \mid \sigma'(v) \neq \emptyset\} \subseteq \{v \text{ Venn region } \mid \sigma(v) \neq \emptyset\}$.

Proof. Using Lemma 3, the proof of Lemma 1 can easily be adapted to QFBAPA$^\infty$.

Regarding decidability in NP, Lemma 3 shows that it is sufficient to prove that solvability in $\mathbb{N}^\infty$ of G together with a polynomially large system $A' \cdot x' = k$

can be decided by an NP procedure. In a first step, we guess which of the PA variables in G and k are to be replaced by ∞. For the linear (in)equations in G containing at least one such variable, the truth value is determined by this choice. For example, if we have $x_1 + 2x_2 > x_3$, and x_1 is guessed to be ∞, then this inequation becomes true if x_3 is not guessed to be ∞, and false otherwise. By replacing such (in)equations by the respective truth values, G can be modified to G', which now needs to be solved in $\mathbb{N}$. Regarding the system $A' \cdot x' = k$, we check whether, for each i such that k_i was guessed to be ∞, there is a Venn region v such that 1. and 2 of Lemma 2 are satisfied, where "$\sigma(k_j) < \infty$" is replaced with "k_j is not guessed to be ∞." If this is not the case, than we return the answer "unsolvable." Otherwise, we modify $A' \cdot x' = k$ to the system $A'' \cdot x' = k'$ by removing the rows i for which k_i was guessed to be ∞. We then check whether G' together with this new system is solvable in $\mathbb{N}$. Using Lemma 2 and the construction employed in the proof of Lemma 3, it is not hard to show that this yields a correct NP procedure. $\square$

In [10], similar results are shown for QFBAPA$_\infty$, but again without the property (ii). From a technical point of view, the proof in [10] is quite different from ours, though it is based on similar ideas.

The logic CQU This logic is obtained from QFBAPA$^\infty$ by restricting numerical constraints to be of the form $k = N$ and $k < N$, i.e., a CQU formula is a Boolean combination of set constraints and numerical constraints of this restricted form. Since CQU is a fragment of QFBAPA$^\infty$, its satisfiability problem is clearly also in NP, and NP hardness is, on the one hand, due to the Boolean operations on the formula level. Other reasons for NP-hardness are the Boolean operations on the set level, and the fact that numerical constraints can be used to express the knapsack problem [14].

It should be noted that the logic CQU as introduced here is actually the *Boolean closure* of the logic called CQU in [7]. In fact, in [7] only conjunctions of set constraints and numerical constraints of the form $k \leq N$ and $k \geq N$ for set cardinalities k are allowed. When using CQU to define our extension $\mathcal{ALCCQU}$ of $\mathcal{ALCQ}$, this difference is irrelevant since the Boolean operations are available anyway on the DL level. In addition, sums in the PA expressions k in $k = N$ and $k < N$ can be reduced away using disjunction (see the next section). It is actually not hard to see that the logic CQU as defined here has the same expressivity as $\mathcal{C}^1$, the one-variable fragment of first-order logic with counting (see, e.g., [15]).

3 The DLs $\mathcal{ALCSCC}^\infty$, $\mathcal{ALCCQU}$, and $\mathcal{ALCQ}t$

In this section, we define the variant $\mathcal{ALCSCC}^\infty$ of the logic $\mathcal{ALCSCC}$ introduced in [2], and its fragments $\mathcal{ALCCQU}$ and $\mathcal{ALCQ}t$. First, we argue why the complexity results for $\mathcal{ALCSCC}$ proved in [2] also hold for $\mathcal{ALCSCC}^\infty$. Then, we show that $\mathcal{ALCCQU}$ has the same expressivity as its fragment $\mathcal{ALCQ}t$, and that both are expressible in first-order logic.

The DL $\mathcal{ALCSCC}^\infty$ Basically, $\mathcal{ALCSCC}^\infty$ provides us with Boolean operations on concepts and constraints on role successors, which are expressed in QFBAPA$^\infty$. In these constraints, role names and concept descriptions can be used as set variables, and there are no PA variables allowed. The syntax of $\mathcal{ALCSCC}^\infty$ is identical to the one of $\mathcal{ALCSCC}$.

Definition 1 (Syntax of $\mathcal{ALCSCC}^\infty$). *Given finite, disjoint sets N_C of concept names and N_R of role names, the set of $\mathcal{ALCSCC}$ concept descriptions over the signature (N_C, N_R) is inductively defined as follows:*

- *every concept name in N_C is an $\mathcal{ALCSCC}$ concept description over (N_C, N_R);*
- *if C, D are $\mathcal{ALCSCC}$ concept descriptions over (N_C, N_R), then so are $C \sqcap D$, $C \sqcup D$, and $\neg C$;*
- *if Con is a set or numerical constraint of QFBAPA$^\infty$ using role names and already defined $\mathcal{ALCSCC}$ concept descriptions over (N_C, N_R) as (set) variables, then $succ(Con)$ is an $\mathcal{ALCSCC}$ concept description over (N_C, N_R).*

An $\mathcal{ALCSCC}$ TBox over (N_C, N_R) is a finite set of concept inclusions of the form $C \sqsubseteq D$, where C, D are $\mathcal{ALCSCC}$ concept descriptions over (N_C, N_R).

For example, the $\mathcal{ALCSCC}^\infty$ concept description $Female \sqcap succ(|child \cap Female| = |child \cap Male|)$ describes all female individuals that have exactly as many sons as daughters. Of course, successor constraints can also be nested, as in the $\mathcal{ALCSCC}^\infty$ concept description $succ(|child \cap succ(child \subseteq Female)| = |child \cap succ(child \subseteq Male)|)$, which describes all individuals having as many children that have only daughters as they have children having only sons. As usual in DL, the semantics of $\mathcal{ALCSCC}^\infty$ is defined using the notion of an interpretation.[3]

Definition 2 (Semantics of $\mathcal{ALCSCC}^\infty$). *Given finite, disjoint sets N_C and N_R of concept and role names, respectively, an* interpretation *of N_C and N_R consists of a non-empty set $\Delta^\mathcal{I}$ and a mapping $\cdot^\mathcal{I}$ that maps every concept name $A \in N_C$ to a subset $A^\mathcal{I}$ of $\Delta^\mathcal{I}$ and every role name $r \in N_R$ to a binary relation $r^\mathcal{I}$ over $\Delta^\mathcal{I}$. Given an individual $d \in \Delta^\mathcal{I}$ and a role name $r \in N_R$, we define $r^\mathcal{I}(d) := \{e \in \Delta^\mathcal{I} \mid (d, e) \in r^\mathcal{I}\}$ (r-successors) and $ars^\mathcal{I}(d) := \bigcup_{r \in N_R} r^\mathcal{I}(d)$ (all role successors).*

The interpretation function $\cdot^\mathcal{I}$ is inductively extended to $\mathcal{ALCSCC}^\infty$ concept descriptions over (N_C, N_R) by interpreting $\sqcap$, $\sqcup$, and $\neg$ respectively as intersection, union and complement. Successor constraints are evaluated according to the semantics of QFBAPA$^\infty$: to determine whether $d \in succ(Con)^\mathcal{I}$ or not, $\mathcal{U}$ is evaluated as $ars^\mathcal{I}(d)$ (i.e., the set of all role successors of d), $\emptyset$ as the empty set, roles r occurring in Con as $r^\mathcal{I}(d)$ (i.e., the set of r-successors of d) and concept descriptions D as $D^\mathcal{I} \cap ars^\mathcal{I}(d)$ (i.e., the set of role successors of d that belong to D).[4] Then $d \in succ(Con)^\mathcal{I}$ iff the substitution obtained this way is a solution of the QFBAPA$^\infty$ formula Con.

[3] A more detailed definition of the semantics can be found in [5].

[4] Note that, by induction, the sets $D^\mathcal{I}$ are well-defined.

The interpretation $\mathcal{I}$ is a model *of the TBox $\mathcal{T}$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ holds for all $C \sqsubseteq D \in \mathcal{T}$. The $\mathcal{ALCSCC}^\infty$ concept description C is* satisfiable *if there is an interpretation $\mathcal{I}$ such that $C^{\mathcal{I}} \neq \emptyset$, and it is* satisfiable w.r.t. the TBox $\mathcal{T}$ *if there is a model $\mathcal{I}$ of $\mathcal{T}$ such that $C^{\mathcal{I}} \neq \emptyset$. The $\mathcal{ALCSCC}^\infty$ concept descriptions C, D are* equivalent *(written $C \equiv D$) if $C^{\mathcal{I}} = D^{\mathcal{I}}$ for all interpretations $\mathcal{I}$.*

This semantics differs from the one given in [2] for $\mathcal{ALCSCC}$ as follows. In [2], interpretations are restricted to being finitely branching in the sense that, for any $d \in \Delta^{\mathcal{I}}$, the set $ars^{\mathcal{I}}(d)$ of all role successors of d must be finite. This ensures that, in the evaluation of successor constraints, only finite sets are considered, and thus this evaluation can be done using QFBAPA. Here, we do not make this assumption, and thus QFBAPA$^\infty$ needs to be used to evaluate successor constraints. Note that in $\mathcal{ALCSCC}^\infty$ we can actually force the existence of elements with infinitely many role successors. For example, the successor constraint $succ(|r| + 1 = |r|)$ is unsatisfiable in $\mathcal{ALCSCC}$, but satisfiable in $\mathcal{ALCSCC}^\infty$ in an interpretation that contains an element that has infinitely many r-successors.

The main results shown in [2] are that the satisfiability problem in $\mathcal{ALCSCC}$ is PSpace-complete for the case without a TBox and ExpTime-complete in the presence of a TBox. The hardness results trivially follow from well-known hardness results for $\mathcal{ALC}$ [17,16]. The main tools used in the proof of the complexity upper bounds are Lemma 1 (and in particular property (ii)) and Proposition 1. Since, by Theorem 1, these two results also hold for QFBAPA$^\infty$, we can basically reuse the proofs from [2]. The only places where explicit adaptations are required are the proofs of soundness of the algorithms, where one now must consider infinite sets. Since these adaptations are quite easy, we dispense with spelling them out here.

Theorem 2. *Satisfiability in $\mathcal{ALCSCC}^\infty$ is PSpace-complete without a TBox and ExpTime-complete in the presence of a TBox.*

The DLs $\mathcal{ALCCQU}$ and $\mathcal{ALCQt}$ Fragments of $\mathcal{ALCSCC}^\infty$ can be obtained by restricting the constraints that can be used in successor constraints to fragments of QFBAPA$^\infty$.

Definition 3 ($\mathcal{ALCCQU}$). *The DL $\mathcal{ALCCQU}$ is defined like $\mathcal{ALCSCC}^\infty$, but in successor constraints only constraints of CQU can be used.*

Thus, the concept description

$$succ(|child \cap livesWith^c \cap Female| = 1), \tag{4}$$

which describes individuals that have exactly one daughter that does not live with them, is an $\mathcal{ALCCQU}$ concept description, but $succ(|child \cap livesWith^c \cap Female| = |child \cap livesWith^c \cap Male|)$ is not. In the definition of CQU given in the previous section, we have introduced as atomic constraints only constraints of the form $k = N$ and $k < N$. In $\mathcal{ALCCQU}$, we can also allow the use of constraints of the form $k \leq N$, $k > N$, $k \geq N$, and $k \neq N$ since successor constraints using them can be expressed. For example, $\mathit{succ}(k \geq N) \equiv \neg \mathit{succ}(k < N)$.

Before we can introduce the fragment $\mathcal{ALCQ}t$ of $\mathcal{ALCCQU}$ we must define the notion of a safe role type. A *role literal* is a role name r or its complement r^c. Given a finite set of role names N_R, a *role type* for N_R is an intersection τ of role literals such that every role name in N_R occurs exactly once in this conjunction. The role type τ is *safe* if at least one role occurs non-negated. For example, if $N_R = \{r, s, t\}$, then $r \cap s \cap t^c$ and $r^c \cap s^c \cap t^c$ are role types, but the latter is not safe. The intersections $r \cap s$ and $r \cap s \cap t^c \cap r^c$ are not role types.

Definition 4 ($\mathcal{ALCQ}t$). *The DL $\mathcal{ALCQ}t$ is defined like $\mathcal{ALCSCC}^\infty$, but in successor constraints occurring in $\mathcal{ALCQ}t$ concept descriptions over (N_C, N_R), no set constraints can be used, and numerical constraints are restricted to the form $|\tau \cap C| \bowtie N$, where τ is a safe role type for N_R, C is an $\mathcal{ALCQ}t$ concept description over (N_C, N_R), and $\bowtie \in \{<, \leq, \geq, >, =, \neq\}$.*

The $\mathcal{ALCCQU}$ concept description (4) is actually an $\mathcal{ALCQ}t$ concept description over (N_C, N_R) if N_R contains only the two roles used in this description.

Adopting the syntax usually employed to denote qualified number restrictions in DLs [1], we can write $\mathcal{ALCQ}t$ successor constraints $\mathit{succ}(|\tau \cap C| \bowtie N)$ as $(\bowtie N\, \tau \,.\, C)$ with τ a safe role type. The semantics given by Definition 2 to the successor constraint $\mathit{succ}(|\tau \cap C| \bowtie N)$ indeed coincides with the usual semantics for qualified number restrictions if intersection and complement in τ are respectively interpreted as role intersection and role complement. It should be noted, however, that this is only true since τ is assumed to be safe. In fact, in Definition 2, complement is performed w.r.t. the role successors of the individual under consideration, whereas general role negation is performed w.r.t. all elements of the interpretation domain. But safety of τ ensures that only role successors of the given individual can be τ-successors of this individual. A similar safety requirement for role expressions has been employed by Tobies (see [19], Chapter 4.4), but he considers arbitrary Boolean combinations of roles, and not just role types, and also allows for inverse roles.

Obviously, $\mathcal{ALCQ}t$ is a sub-logic of $\mathcal{ALCCQU}$ since the successor constraints available in $\mathcal{ALCQ}t$ can clearly be expressed in $\mathcal{ALCCQU}$. From a syntactic point of view, $\mathcal{ALCCQU}$ has successor constraints that are not available in $\mathcal{ALCQ}t$. However, we can show that nevertheless all of them can be expressed in $\mathcal{ALCQ}t$.

Theorem 3. *The DLs $\mathcal{ALCCQU}$ and $\mathcal{ALCQ}t$ have the same expressivity.*

Proof. We need to show that the successor constraints of $\mathcal{ALCCQU}$ can be expressed in $\mathcal{ALCQ}t$. Because of space restrictions, we refer the reader to [5] for a detailed proof, and only give a sketch here.

First note that set constraints, which can be used in $\mathcal{ALCCQU}$, but not in $\mathcal{ALCQ}t$, can be expressed as numerical constraints. Indeed, we have $\mathit{succ}(s \subseteq t) \equiv \mathit{succ}(|s \cap t^c| = 0)$. Second, numerical constraints in $\mathcal{ALCCQU}$ may contain linear combinations of set cardinalities, whereas addition and multiplication with a constant are not allowed to be used in numerical constraints of $\mathcal{ALCQ}t$. However, they can be eliminated using Boolean operations. In fact, multiplication with a non-negative integer constant can be expressed by iterated addition, and addition can be eliminated as follows: $succ(|s_1| + \ldots + |s_\ell| \bowtie N)$ for $\bowtie \in \{\leq, =, \geq\}$ is equivalent to the disjunction

$$\bigsqcup_{N_1+\ldots+N_\ell=N} succ(|s_1| \bowtie N_1) \sqcap \ldots \sqcap succ(|s_\ell| \bowtie N_\ell),$$

where $N_1, \ldots, N_\ell$ range over the non-negative integers. Since N is a fixed non-negative integer, this disjunction is clearly finite. For $\bowtie \in \{<, >\}$, this equivalence would not hold, but we can clearly express $>$ and $<$ using the other comparison operators. For example, $succ(k > N) \equiv \neg succ(k \leq N)$.

Finally, consider successor constraints of the form $succ(|s| \bowtie N)$ where s is a set term built using role names and concept descriptions, and $\bowtie \in \{\leq, =, \geq\}$. The semantics of $\mathcal{ALCSCC}^\infty$ ensures that $succ(|s| \bowtie N)$ is equivalent to $succ(|s \cap (r_1 \cup \ldots \cup r_n)| \bowtie N)$ where $N_R = \{r_1, \ldots, r_n\}$. Using distributivity of set intersection over set union, we can now transform the set term $s \cap (r_1 \cup \ldots \cup r_n)$ into "disjunctive normal form," which yields an equivalent set term of the form $(s_1 \cap C_1) \cup \ldots \cup (s_p \cap C_p)$, where each s_i is a conjunction of role literals containing at least one role positively, and the C_i are concept descriptions. Obviously, each s_i can then be expressed as a union of safe role types. Using the fact that $(\tau \cap C_1) \cup (\tau \cap C_2)$ is equivalent to $(\tau \cap (C_1 \sqcup C_2))$, we thus obtain that $s \cap (r_1 \cup \ldots \cup r_n)$ can be expressed in the form

$$(\tau_1 \cap D_1) \cup \ldots \cup (\tau_q \cap D_q),$$

where $\tau_1, \ldots, \tau_q$ are distinct safe role types and $D_1, \ldots, D_q$ are concept descriptions. Since distinct role types are interpreted as disjoint sets, we thus have

$$\begin{aligned} succ(|s| \bowtie N) &\equiv succ(|(\tau_1 \cap D_1)| + \ldots + |(\tau_q \cap D_q)| \bowtie N) \\ &\equiv \bigsqcup_{N_1+\ldots+N_q=N} succ(|\tau_1 \cap D_1| \bowtie N_1) \sqcap \ldots \sqcap succ(|\tau_q \cap D_q| \bowtie N_q). \end{aligned}$$

Since the last expression is clearly an $\mathcal{ALCQ}t$ concept description, this completes the proof of the theorem. □

It should be noted that the constructions employed in the above proof can lead to an exponential blow-up for several reasons. One example is building the disjunctive normal form in the third part of the proof, and another is expressing addition using disjunction. Thus, it would not be a good idea to use these constructions for the purpose of reducing reasoning in $\mathcal{ALCCQU}$ to reasoning in $\mathcal{ALCQ}t$. At the moment, it is not clear to us whether the exponential blow-up can be avoided, but we conjecture that $\mathcal{ALCCQU}$ is exponentially more succinct than $\mathcal{ALCQ}t$.

By using the standard translation of $\mathcal{ALCQ}$ and of Boolean operations on roles into first-order logic [4], it can easily be shown that $\mathcal{ALCQ}t$, and thus also $\mathcal{ALCCQU}$, can be expressed in first-order logic.

Corollary 1. *The DL $\mathcal{ALCCQU}$ can be expressed in first-order logic, i.e., for every $\mathcal{ALCCQU}$ concept description C there is an* equivalent *first-order formula, i.e, a first-order formula $\phi_C(x)$ with one free variable x such that $C^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} \mid \mathcal{I} \models \phi_C(d)\}$ holds for every interpretation $\mathcal{I}$.*

4 Expressive power

To prove that the concepts of one DL can be expressed in another DL, one usually shows how to construct, for a given concept of the former DL, an equivalent one of the latter, as we have, e.g., done in the proof of Theorem 3. Showing inexpressibility results is usually more involved. An important tool often used in this context is the notion of a bisimulation [12,13], inherited from modal logics [20]. In the following, we first recall the definition of a bisimulation relation tailored towards the DL $\mathcal{ALCQ}$ [13], and use this to show that $\mathcal{ALCCQU}$ is strictly more expressive than $\mathcal{ALCQ}$. Then, we adapt this definition to obtain a bisimulation relation tailored towards $\mathcal{ALCQ}t$. Surprisingly, this relation cannot be used to separate $\mathcal{ALCQ}t$, and thus $\mathcal{ALCCQU}$, from $\mathcal{ALCSCC}^\infty$. In fact, we can show that not only all $\mathcal{ALCCQU}$ concepts are invariant under this notion of bisimulation, but also all $\mathcal{ALCSCC}^\infty$ concepts. As a consequence, we obtain that $\mathcal{ALCCQU}$ is exactly the first-order fragment of $\mathcal{ALCSCC}^\infty$. Finally, we show that $\mathcal{ALCSCC}^\infty$ indeed contains concepts that are not expressible in first-order logic.

$\mathcal{ALCQ}$ bisimulation In the context of the present paper, $\mathcal{ALCQ}$ can be defined as the fragment of $\mathcal{ALCCQU}$ where only successor constraints of the form $succ(|r \cap C| \bowtie N)$ can be used, where $r \in N_R$, C is an $\mathcal{ALCQ}$ concept description, $\bowtie \in \{<, \leq, \geq, >, =, \neq\}$, and N is a non-negative integer.

Definition 5 ([13]). *Let $\mathcal{I}_1$ and $\mathcal{I}_2$ be interpretations. The relation $\rho \subseteq \Delta^{\mathcal{I}_1} \times \Delta^{\mathcal{I}_2}$ is an $\mathcal{ALCQ}$* bisimulation *between $\mathcal{I}_1$ and $\mathcal{I}_2$ if*

1. *$d_1\ \rho\ d_2$ implies $d_1 \in A^{\mathcal{I}_1}$ iff $d_2 \in A^{\mathcal{I}_2}$, for all $d_1 \in \Delta^{\mathcal{I}_1}$, $d_2 \in \Delta^{\mathcal{I}_2}$, $A \in N_C$;*
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 $d_1 \in \Delta^{\mathcal{I}_1}$, $d_2 \in \Delta^{\mathcal{I}_2}$ are $\mathcal{ALCQ}$ bisimilar *(written $(\mathcal{I}_1, d_1) \sim_{\mathcal{ALCQ}} (\mathcal{I}_2, d_2)$) if there is an $\mathcal{ALCQ}$ bisimulation ρ between $\mathcal{I}_1$ and $\mathcal{I}_2$ such that $d_1\ \rho\ d_2$, and* $\mathcal{ALCQ}$-equivalent *(written $(\mathcal{I}_1, d_1) \equiv_{\mathcal{ALCQ}} (\mathcal{I}_2, d_2)$) if for all $\mathcal{ALCQ}$ concept descriptions C we have $d_1 \in C^{\mathcal{I}_1}$ iff $d_2 \in C^{\mathcal{I}_2}$.*

The following theorem shows that $\mathcal{ALCQ}$ is exactly the fragment of first-order logic that is invariant under this notion of bisimulation. We say that a first-order formula $\phi(x)$ with one free variable x is *invariant under* $\sim_{\mathcal{ALCQ}}$ if $(\mathcal{I}_1, d_1) \sim_{\mathcal{ALCQ}} (\mathcal{I}_2, d_2)$ implies $\mathcal{I}_1 \models \phi(d_1)$ iff $\mathcal{I}_2 \models \phi(d_2)$.

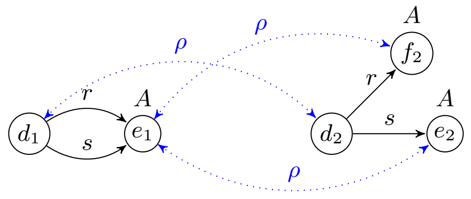

Fig. 1. Two interpretations $\mathcal{I}_1$ and $\mathcal{I}_2$ and an $\mathcal{ALCQ}$ bisimulation ρ.

Theorem 4 ([13]). *Let $\phi(x)$ be a first-order formula with one free variable x. Then the following are equivalent:*

1. *there is an $\mathcal{ALCQ}$ concept description C such that C is equivalent to $\phi(x)$;*
2. *$\phi(x)$ is invariant under $\sim_{\mathcal{ALCQ}}$.*

Since $\mathcal{ALCQ}$ is expressible in first-order logic, this theorem in particular implies that $\mathcal{ALCQ}$ bisimilar elements of interpretations are also $\mathcal{ALCQ}$-equivalent. We can use this fact to show that $\mathcal{ALCCQU}$ is strictly more expressive than $\mathcal{ALCQ}$.

Corollary 2. *There is no $\mathcal{ALCQ}$ concept description C such that C is equivalent to the $\mathcal{ALCCQU}$ concept description $succ(|r \cap s| > 0)$.*

Proof. Assume that C is an $\mathcal{ALCQ}$ concept description such that $C \equiv succ(|r \cap s| > 0)$, and consider the interpretations $\mathcal{I}_1, \mathcal{I}_2$ and the $\mathcal{ALCQ}$ bisimulation ρ depicted in Fig. 1. Then we have $d_1 \in succ(|r \cap s| > 0)^{\mathcal{I}_1} = C^{\mathcal{I}_1}$, and thus $(\mathcal{I}_1, d_1) \sim_{\mathcal{ALCQ}} (\mathcal{I}_2, d_2)$ implies $d_2 \in C^{\mathcal{I}_2}$. This contradicts our assumption that $C \equiv succ(|r \cap s| > 0)$ since $d_2 \notin succ(|r \cap s| > 0)$. □

$\mathcal{ALCQ}t$ bisimulation The definition of a bisimulation can be adapted to $\mathcal{ALCQ}t$ by replacing role names r with safe role types τ. To be more precise, let τ be a safe role type, $r_1, \ldots, r_k$ the role names occurring positively in τ, and $s_1, \ldots, s_\ell$ the role names occurring negatively, i.e., $\tau = r_1 \cap \ldots \cap r_k \cap s_1^c \cap \ldots \cap s_\ell^c$. For a given interpretation $\mathcal{I}$ and element $d \in \Delta^{\mathcal{I}}$ we then define

$$\tau^{\mathcal{I}}(d) := (r_1^{\mathcal{I}}(d) \cap \ldots \cap r_k^{\mathcal{I}}(d)) \setminus (s_1^{\mathcal{I}}(d) \cup \ldots \cup s_\ell^{\mathcal{I}}(d)).$$

Since τ is safe, we have $k \geq 1$, and thus $\tau^{\mathcal{I}}(d) \subseteq r_1^{\mathcal{I}}(d) \subseteq ars^{\mathcal{I}}(d)$.

$\mathcal{ALCQ}t$ bisimulation, *$\mathcal{ALCQ}t$ bisimilar*, *$\mathcal{ALCQ}t$ equivalent*, and *invariance* under $\mathcal{ALCQ}t$ bisimulation are now defined as for $\mathcal{ALCQ}$ (Definition 5), but with $\mathcal{ALCQ}t$ replacing $\mathcal{ALCQ}$ and safe role types τ for N_R replacing role names $r \in N_R$ (see [5] for a more detailed definition).

Theorem 5. *Let $\phi(x)$ be a first-order formula with one free variable x. Then the following are equivalent:*

1. *there is an $\mathcal{ALCQ}t$ concept description C such that C is equivalent to $\phi(x)$;*

2. *$\phi(x)$ is invariant under $\sim_{\mathcal{ALCQ}t}$.*

Since the proof of this theorem is very similar to the proof of Theorem 4 given in [13], we omit it here. An explicit proof for the case of $\mathcal{ALCQ}t$, which is more detailed than the one in [13], can be found in [5].

The expressivity of $\mathcal{ALCSCC}^\infty$ relative to $\mathcal{ALCCQU}$ Our original expectation was that we could use Theorem 5 to show that $\mathcal{ALCSCC}^\infty$ is strictly more expressive than $\mathcal{ALCQ}t$, and thus also $\mathcal{ALCCQU}$. The following proposition implies that this is not possible.

Proposition 2. *If $(\mathcal{I}_1, d_1) \sim_{\mathcal{ALCQ}t} (\mathcal{I}_2, d_2)$ then $(\mathcal{I}_1, d_1) \equiv_{\mathcal{ALCSCC}^\infty} (\mathcal{I}_2, d_2)$.*

A detailed proof of this proposition can be found in [5]. Here, we only explain the main ideas underlying this proof. Basically, we must show that, given a PA expression k occurring in an $\mathcal{ALCSCC}^\infty$ concept description, the assumption that $(\mathcal{I}_1, d_1) \sim_{\mathcal{ALCQ}t} (\mathcal{I}_2, d_2)$ implies that evaluating k on the role successors of d_1 yields the same result (i.e., element of $\mathbb{N}^\infty$) as evaluating k on the role successors of d_2. To this purpose, we first observe that k can be written as

$$k = \sum_{i=1}^{\ell} N_i \cdot |\tau_i \cap C_i|,$$

where (for $i = 1, \ldots, \ell$) we have that N_i is a non-negative integer, τ_i is a safe role type, and C_i is an $\mathcal{ALCSCC}^\infty$ concept description. This can be shown by a simple adaptation of the arguments used in the proof of Theorem 3. Consequently, it is sufficient to show the claim for the summands $|\tau_i \cap C_i|$. First, assume that, on the role successors of d_1, this expression evaluates to the non-negative integer N. Then we have $d_1 \in succ(|\tau_i \cap C_i| = N)^{\mathcal{I}_1}$, and since $succ(|\tau_i \cap C_i| = N)$ is an $\mathcal{ALCQ}t$ concept description, Theorem 5 yields $d_2 \in succ(|\tau_i \cap C_i| = N)^{\mathcal{I}_2}$. Consequently, $|\tau_i \cap C_i|$ also evaluates to N on the role successors of d_2. If $|\tau_i \cap C_i|$ evaluates to ∞ on the role successors of d_1, then we have $d_1 \in succ(|\tau_i \cap C_i| > N)^{\mathcal{I}_1}$ for all non-negative integers N, and we can conclude that $d_2 \in succ(|\tau_i \cap C_i| > N)^{\mathcal{I}_2}$ for all non-negative integers N. This shows that $|\tau_i \cap C_i|$ also evaluates to ∞ on the role successors of d_2, which concludes our proof sketch.

Together with Theorem 5, this proposition yields a characterization of $\mathcal{ALCCQU}$ as the first-order fragment of $\mathcal{ALCSCC}^\infty$.

Theorem 6. *Let C be an $\mathcal{ALCSCC}^\infty$ concept description. Then the following are equivalent:*

1. *there is a first-order formula $\phi(x)$ with one free variable x such that C is equivalent to $\phi(x)$;*
2. *C is equivalent to an $\mathcal{ALCCQU}$ concept description.*

Proof. ($2 \Rightarrow 1$) is an immediate consequence of Corollary 1. Now, assume that 1. holds. Since $\phi(x)$ is equivalent to an $\mathcal{ALCSCC}^\infty$ concept description, it is invariant under $\mathcal{ALCQ}t$ bisimulation by Proposition 2, and thus equivalent to an $\mathcal{ALCQ}t$ concept description by Theorem 5. Since $\mathcal{ALCQ}t$ is a sub-logic of $\mathcal{ALCCQU}$, this yields 2. □

It remains to show that $\mathcal{ALCSCC}^\infty$ itself is not a fragment of first-order logic.

Theorem 7. *The $\mathcal{ALCSCC}^\infty$ concept description $succ(|r \cap A| = |r \cap \neg A|)$ cannot be expressed in first-order logic.*

To prove this theorem, it is sufficient to show that the above concept description cannot be expressed in $\mathcal{ALCCQU}$. The proof of this fact is similar to the one given in [2] to show that $\mathcal{ALCSCC}$ is more expressive than $\mathcal{ALCQ}$, but more involved since $\mathcal{ALCCQU}$ is more expressive that $\mathcal{ALCQ}$.

5 Conclusion

In this paper, we have introduced the variant $\mathcal{ALCSCC}^\infty$ of the DL $\mathcal{ALCSCC}$ investigated in [2], in which the restriction to finitely branching interpretations is lifted. We have shown that this modification does not change the complexity of reasoning. As an auxiliary result we have shown that reasoning in QFBAPA$^\infty$, a variant of QFBAPA in which also infinite sets are allowed, has the same complexity as in QFBAPA. The main result of this paper is the proof that the DL $\mathcal{ALCCQU}$ is exactly the first-order fragment of $\mathcal{ALCSCC}^\infty$.

Regarding future work, it should be noted that we have only investigated the expressive power of the *concept descriptions* of $\mathcal{ALCSCC}^\infty$ and $\mathcal{ALCCQU}$. In [13], the expressivity of *TBoxes* is considered as well. It would be interesting to see whether we can extend our results to TBoxes (or even cardinality constraints on concepts [3]) of $\mathcal{ALCSCC}^\infty$ and $\mathcal{ALCCQU}$. In [5] we have shown how the satisfiability procedure for CQU presented in [7], which is based on column generation, can be extended to a satisfiability procedure for $\mathcal{ALCCQU}$, but it remains to implement and test this procedure. In addition, it would be interesting to see whether this approach can be extended to $\mathcal{ALCSCC}$ and $\mathcal{ALCSCC}^\infty$. It would also be interesting to see what impact the addition of inverse roles to $\mathcal{ALCSCC}$ $\mathcal{ALCSCC}^\infty$ has on the complexity of reasoning.

Acknowledgment The authors would like to thank Ulrike Baumann for helpful discussions regarding QFBAPA$^\infty$. We should also like to point out that we have learned about the results regarding QFBAPA$_\infty$ in [10] only a couple of days before the submission of the final version of this paper.

References

1. Franz Baader. Description logic terminology. In *[4]*, pp. 485–495. 2003.
2. Franz Baader. A new description logic with set constraints and cardinality constraints on role successors. In *Proc. of the 11th Int. Symp. on Frontiers of Combining Systems (FroCoS'17)*, Springer LNCS 10483, 2017.

3. Franz Baader. Expressive cardinality constraints on $\mathcal{ALCSCC}$ concepts. In *Proc. of the 34th Annual ACM Symposium on Applied Computing (SAC'19)*, pp. 1123–1131. ACM, 2019.
4. Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications.* Cambridge University Press, 2003.
5. Filippo De Bortoli. Integrating reasoning services for description logics with cardinality constraints with numerical optimization techniques. EMCL Master's Thesis, Chair for Automata Theory, Faculty of Computer Science, TU Dresden, 2019. Available online at https://tu-dresden.de/inf/lat/theses#DeBo-Mas-19.
6. Friedrich Eisenbrand and Gennady Shmonin. Carathéodory bounds for integer cones. *Oper. Res. Lett.*, 34(5):564–568, 2006.
7. Marcelo Finger and Glauber De Bona. Algorithms for deciding counting quantifiers over unary predicates. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI'17)*, pp. 3878–3884. AAAI Press, 2017.
8. Robert Hoehndorf, Paul N. Schofield, and Georgios V. Gkoutos. The role of ontologies in biological and biomedical research: A functional perspective. *Brief. Bioinform.*, 16(6):1069–1080, 2015.
9. Bernhard Hollunder and Franz Baader. Qualifying number restrictions in concept languages. In *Proc. of the 2nd Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR'91)*, pp. 335–346, 1991.
10. Viktor Kuncak, Ruzica Piskac, and Philippe Suter. Ordered sets in the calculus of data structures. In *Computer Science Logic*, Springer LNCS 6427, pp. 34–48. 2010.
11. Viktor Kuncak and Martin C. Rinard. Towards efficient satisfiability checking for Boolean algebra with Presburger arithmetic. In *Proc. of the 21st Int. Conf. on Automated Deduction (CADE-07)*, Springer LNCS 4603, pp. 215–230. 2007.
12. Natasha Kurtonina and Maarten de Rijke. Expressiveness of concept expressions in first-order description logics. *Artificial Intelligence*, 107(2):303–333, 1999.
13. Carsten Lutz, Robert Piro, and Frank Wolter. Description logic TBoxes: Model-theoretic characterizations and rewritability. In *Proc. of the 22nd Int. Joint Conf. on Artificial Intelligence (IJCAI'11)*, pp. 983–988. IJCAI/AAAI, 2011.
14. Christos H. Papadimitriou. On the complexity of integer programming. *J. of the ACM*, 28(4):765–768, 1981.
15. Ian Pratt-Hartmann. On the computational complexity of the numerically definite syllogistic and related logics. *Bulletin of Symbolic Logic*, 14(1):1–28, 2008.
16. Klaus Schild. A correspondence theory for terminological logics: Preliminary report. In *Proc. of the 12th Int. Joint Conf. on Artificial Intelligence (IJCAI'91)*, pp. 466–471, 1991.
17. Manfred Schmidt-Schauß and Gert Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48(1):1–26, 1991.
18. Stephan Tobies. A PSPACE algorithm for graded modal logic. In *Proc. of the 16th Int. Conf. on Automated Deduction (CADE'99)*, Springer LNAI 1632, pp. 52–66. 1999.
19. Stephan Tobies. *Complexity Results and Practical Algorithms for Logics in Knowledge Representation.* PhD thesis, LuFG Theoretical Computer Science, RWTH-Aachen, Germany, 2001. http://tu-dresden.de/inf/lat/theses/#Tobies-PhD-2001.
20. Johan van Benthem. *Modal Logic and Classical Logic.* Bibliopolis, Napoli, 1983.