RWTH
LTCS–Report

Aachen University of Technology
Research group for
Theoretical Computer Science

Description Logics with Aggregates and Concrete Domains, Part II (extended)

Franz Baader Ulrike Sattler

LTCS-Report 98-02

This is a new, extended version of a report with the same number.

An abridged version has appeared in the Proceedings of the European Conference on Artificial Intelligence, Brighton, UK, 1998.

RWTH Aachen
LuFg Theoretische Informatik
http://www-lti.informatik.rwth-aachen.de

Ahornstr. 55
52074 Aachen
Germany

Description Logics with Aggregates and Concrete Domains, Part II (extended)

Franz Baader Ulrike Sattler

Abstract

We extend different Description Logics by concrete domains (such as integers and reals) and by aggregation functions over these domains (such as min, max, count, sum), which are usually available in database systems. We present decision procedures for the inference problems satisfiability for these Logics—provided that the concrete domain is not too expressive. An example of such a concrete domain is the set of (nonnegative) integers with comparisons ($=$, $\leq$, $\leq_n$, ...) and the aggregation functions min, max, count.

1 Motivation

Unlike many other expressive representation formalisms, such as database schema and query languages, basic Description Logic formalisms (e.g., $\mathcal{ALC}$ [Schmidt-Schauß & Smolka,1991; Donini *et al.*,1991]) do not allow for built-in predicates (like comparisons of numbers) and for aggregation functions (like sum, min, max, average, count). The first deficit was overcome in [Baader & Hanschke,1991], where a generic extension of $\mathcal{ALC}$ by a *concrete domain* $\mathcal{D}$ was proposed. In this extended DL, called $\mathcal{ALC}(\mathcal{D})$, abstract individuals (which are described using $\mathcal{ALC}$) can be related to values in the *concrete domain* $\mathcal{D}$ (e.g., the integers, strings, ...) via so-called *features*, i.e., functional roles. This allows one to describe, for example, managers that spend more money than they earn by

$$\texttt{Manager} \sqcap (\mathsf{less}(\texttt{income}, \texttt{expenses})).$$

In our extension of $\mathcal{ALC}(\mathcal{D})$, aggregation is viewed as a means to define new features. Figure 1 describes a situation where the income and expenses of a person, `Josie`, are given per month. In some months, she spends more money than she earns, and in others less. If we want to know the difference between income and expenses over a whole year, we must be able to build the sum over all months. Then we can state that, or ask whether, `Josie` is an instance of

$$\texttt{Human} \sqcap (\exists \texttt{year}.\texttt{less}(\,\texttt{sum}(\texttt{month} \circ \texttt{income}),\; \texttt{sum}(\texttt{month} \circ \texttt{expenses}))),$$

where the complex feature $\texttt{sum}(\texttt{month} \circ \texttt{income})$ relates an individual to the sum over all values reachable over `month` followed by `income`. This new, complex feature is built using the aggregation function `sum`, the role name `month`, and the feature `income`.

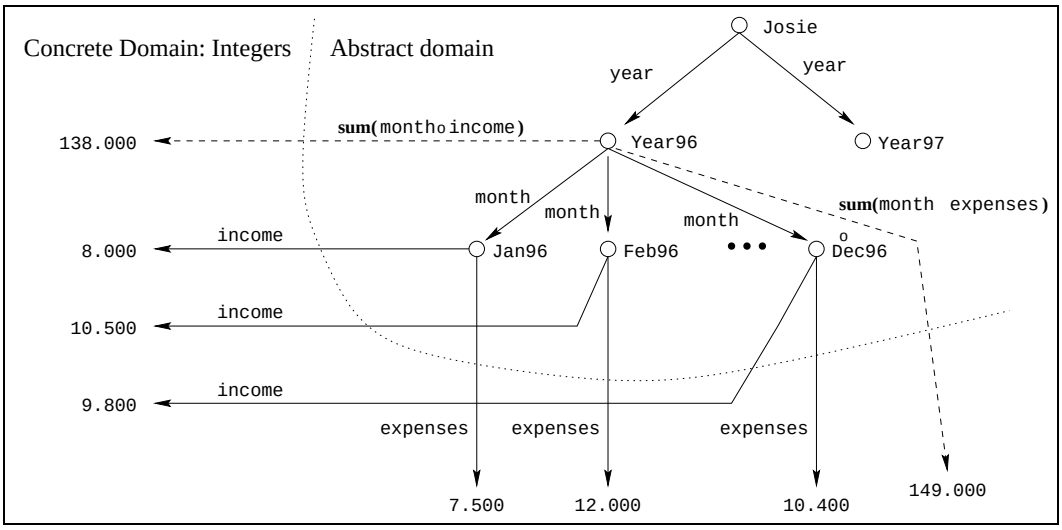

Figure 1: Example for aggregation

In this paper, we present a generic extension of $\mathcal{ALC}(\mathcal{D})$ by aggregation that is based on this idea of introducing new "aggregated features". Unfortunately, it turned out that, given a concrete domain together with aggregation functions satisfying some rather weak conditions, this extension has an undecidable satisfiability and subsumption problem [Baader & Sattler,1997].

Moreover, this result can even be tightened: extending $\mathcal{FL}_0$, a very weak Description Logic allowing for conjunction and *universal* value restrictions only, by aggregation already causes undecidability.

This high complexity of the relevant inference problems is due to the interaction between universal value restrictions and aggregation functions. To obtain decidable Description Logics with aggregation, $\mathcal{CQ}(\mathcal{D}_{\mathsf{agg}})$, a restriction of $\mathcal{ALC}(\mathcal{D}_{\mathsf{agg}})$ obtained by disallowing universal value restriction and restricting negation to concept names, is defined. In this paper, we present a tableau-based algorithm that decides satisfiability of $\mathcal{CQ}(\mathcal{D}_{\mathsf{agg}})$-concepts, provided that satisfiability of finite conjunctions of concrete predicates involving aggregations on multiset variables in the concrete domain $\mathcal{D}$ can be decided. For example, the (nonnegative) integers, rational or real numbers with comparisons $>, \geq, \ldots$, possibly involving constants, together with the aggregation function $\mathsf{min}, \mathsf{max}, \mathsf{count}$ belong to the concrete domains for which the satisfiability of these conjunctions can be decided.

2 The basic Description Logic $\mathcal{ALC}(\mathcal{D})$

In this section, $\mathcal{ALC}(\mathcal{D})$, the Description Logic underlying this investigation, is presented. $\mathcal{ALC}(\mathcal{D})$ is an extension of the well-known Description Logic $\mathcal{ALC}$ (see [Schmidt-Schauß & Smolka,1991; Hollunder *et al.*,1990; Donini *et al.*,1991; 1995]) by so-called *concrete domains*. First, we formally specify a concrete domain.

Definition 1 (Concrete Domains)
A *concrete domain* $\mathcal{D} = (\mathsf{dom}(\mathcal{D}), \mathsf{pred}(\mathcal{D}))$ consists of

- a set $\mathsf{dom}(\mathcal{D})$ (the domain), and
- a set of predicate symbols $\mathsf{pred}(\mathcal{D})$.

Each predicate symbol $P \in \mathsf{pred}(\mathcal{D})$ is associated with an arity n and an n-ary relation $P^{\mathcal{D}} \subseteq \mathsf{dom}(\mathcal{D})^n$.

In [Baader & Hanschke,1991], concrete domains are restricted to so-called *admissible* concrete domains in order to keep the inference problems of this extension decidable. We recall that, roughly spoken, a concrete domain $\mathcal{D}$ is called *admissible* iff (1) $\mathsf{pred}(\mathcal{D})$ is closed under negation and contains a unary predicate name $\top_{\mathcal{D}}$ for $\mathsf{dom}(\mathcal{D})$, and (2) satisfiability of finite conjunctions over $\mathsf{pred}(\mathcal{D})$ is decidable.

The syntax of $\mathcal{ALC}(\mathcal{D})$-concepts is defined as follows (see [Baader & Hanschke,1991]):

Definition 2 Let N_C, N_R, and N_F be disjoint sets of *concept*, *role*, and *feature names*. The set of $\mathcal{ALC}(\mathcal{D})$-*concepts* is the smallest set such that

1. every concept name is a concept and
2. if C, D are concepts, R is a role or a feature name, $P \in \mathsf{pred}(\mathcal{D})$ is an n-ary predicate name, and $u_1, \ldots, u_n$ are feature chains,[1] then $(C \sqcap D)$, $(C \sqcup D)$, $(\neg C)$, $(\forall R.C)$, $(\exists R.C)$, and $P(u_1, \ldots, u_n)$ are concepts.

Concepts of the form $P(u_1, \ldots, u_n)$ are called *predicate restrictions*, and concepts of the form $(\forall R.C)$ (resp. $(\exists R.C)$) are called *universal* (resp. *existential*) *value restrictions*. In order to fix the exact meaning of these concepts, their semantics is defined in the usual model-theoretic way.

Definition 3 An *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of a set $\Delta^{\mathcal{I}}$ disjoint from $\mathsf{dom}(\mathcal{D})$, called the *domain* of $\mathcal{I}$, and a function $\cdot^{\mathcal{I}}$ which maps every concept to a subset of $\Delta^{\mathcal{I}}$, every role to a subset of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, and every feature name $f \in N_F$ to a partial function $f^{\mathcal{I}} : \Delta^{\mathcal{I}} \to \Delta^{\mathcal{I}} \cup \mathsf{dom}(\mathcal{D})$. Furthermore, $\mathcal{I}$ has to satisfy the following properties

$$
\begin{aligned}
(C \sqcap D)^{\mathcal{I}} &= C^{\mathcal{I}} \cap D^{\mathcal{I}} \\
(C \sqcup D)^{\mathcal{I}} &= C^{\mathcal{I}} \cup D^{\mathcal{I}} \\
\neg C^{\mathcal{I}} &= \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} \\
(\exists R.C)^{\mathcal{I}} &= \{d \in \Delta^{\mathcal{I}} \mid \text{Exists } e \in \Delta^{\mathcal{I}} \text{ with } (d,e) \in R^{\mathcal{I}} \text{ and } e \in C^{\mathcal{I}}\} \\
(\forall R.C)^{\mathcal{I}} &= \{d \in \Delta^{\mathcal{I}} \mid \text{For all } e \in \Delta^{\mathcal{I}}, \text{ if } (d,e) \in R^{\mathcal{I}}, \text{ then } e \in C^{\mathcal{I}}\} \\
P(u_1, \ldots, u_n)^{\mathcal{I}} &= \{a \in \Delta^{\mathcal{I}} \mid (u_1^{\mathcal{I}}(a), \ldots, u_n^{\mathcal{I}}(a)) \in P^{\mathcal{D}}\}
\end{aligned}
$$

[1] A feature chain $u = f_1 \circ \ldots \circ f_m$ is a sequence of features f_i.

where $(f_1 \circ \ldots \circ f_m)^{\mathcal{I}}(a) = f_m^{\mathcal{I}}(f_{m-1}^{\mathcal{I}}(\ldots(f_1^{\mathcal{I}}(a)\ldots)$. A concept C is called *satisfiable* iff there is some interpretation $\mathcal{I}$ such that $C^{\mathcal{I}} \neq \emptyset$. Such an interpretation is called a *model* of C. A concept D *subsumes* a concept C (written $C \sqsubseteq D$) iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ holds for each interpretation $\mathcal{I}$.

For an interpretation $\mathcal{I}$, an individual $a \in \Delta^{\mathcal{I}}$ is called an *instance* of a concept C iff $a \in C^{\mathcal{I}}$. An individual $b \in \Delta^{\mathcal{I}}$ is said to be an R-successor of $a \in \Delta^{\mathcal{I}}$ iff $(a, b) \in R^{\mathcal{I}}$. Similar, $\ell \in \Delta^{\mathcal{I}} \cup \mathsf{dom}(\mathcal{D})$ is an $f_1 \ldots f_n$-successor of $a \in \Delta^{\mathcal{I}}$ iff $f_n^{\mathcal{I}}(f_{n-1}^{\mathcal{I}}(\ldots(f_1^{\mathcal{I}}(a)\ldots) = \ell$.

As a consequence of this definition, an instance of a concept $P(u_1, \ldots, u_n)$ has necessarily an u_i-successor in $\mathsf{dom}(\mathcal{D})$ for each $1 \leq i \leq n$. Furthermore, if $x \in \top_{\mathcal{D}}(f)^{\mathcal{I}}$, then $f^{\mathcal{I}}(x) \in \mathsf{dom}(\mathcal{D})$. As $\mathcal{ALC}(\mathcal{D})$ allows for negation and conjunction of concepts, all boolean operators can be expressed. Another consequence of the presence of these two operators is that subsumption and (un)satisfiability can be reduced to each other:

- $C \sqsubseteq D$ iff $C \sqcap \neg D$ is unsatisfiable,
- C is unsatisfiable iff $C \sqsubseteq A \sqcap \neg A$ (for some concept name A).

From the results presented in [Baader & Hanschke,1991] it follows immediately that subsumption and satisfiability are decidable for $\mathcal{ALC}(\mathcal{D})$ concepts—given that $\mathcal{D}$ is admissible. The authors present a tableau-based procedure that decides these and other inference problems.

3 Extension of $\mathcal{ALC}(\mathcal{D})$ by aggregation

In order to define aggregation appropriately, first, we will introduce the notion of *multisets*: In contrast to simple sets, in a multiset an individual can occur more than once; for example, the multiset $\{\{1\}\}$ is different from the multiset $\{\{1, 1\}\}$. Multisets are needed to ensure, e.g., that `Josie`'s income is calculated correctly in the case she earns the same amount of money in more than one month.

Definition 4 (Multisets) Let S be a set. A *multiset* M over S is a mapping $M : S \to \mathbb{N}$, where $M(s)$ denotes the number of occurrences of s in M. The set of all multisets of S is denoted by $\mathsf{MS}(S)$. We write $s \in M$ for $M(s) \geq 1$, and $M \subseteq M'$ for $M(s) \leq M'(s)$ for all $s \in S$. A multiset M is said to be finite iff $\{s \mid M(s) \neq 0\}$ is a finite set. To enumerate the members a_i of a finite multiset, we use the notation $\{\{a_1, \ldots, a_n\}\}$ to distinguish multisets from sets.

As the aggregation functions strongly depend on the specific concrete domains, the notion of a *concrete domain* is extended accordingly. Furthermore, the notion of *concrete features* is introduced. Those are (possibly complex) features which can be built using aggregation over roles followed by features. Then $\mathcal{ALC}(\mathcal{D}_{\mathsf{agg}})$-concepts are defined.

Definition 5 The notion of a concrete domain $\mathcal{D}$ as introduced in Definition 1 is extended by a set of aggregation functions $\mathsf{agg}(\mathcal{D})$, where each $\Sigma \in \mathsf{agg}(\mathcal{D})$ is associated with a partial function $\Sigma^{\mathcal{D}}$ from the set of multisets of $\mathsf{dom}(\mathcal{D})$ into $\mathsf{dom}(\mathcal{D})$.

Let $f, f_1, f_2 \ldots \in N_F$, $R \in N_R$, and $\Sigma \in \mathsf{agg}(\mathcal{D})$. The set of *concrete features* is defined as follows:

- A feature chain $f_1 \ldots f_n$ is a concrete feature, and
- an aggregated feature $f_1 \ldots f_n \circ \Sigma(R \circ f)$ is a concrete feature.

Finally, $\mathcal{ALC}(\mathcal{D}_{\mathsf{agg}})$-concepts are obtained from $\mathcal{ALC}(\mathcal{D})$-concepts by allowing, additionally, the use of concrete features u_i in a predicate restrictions $P(u_1, \ldots, u_n)$ (recall that in $\mathcal{ALC}(\mathcal{D})$ only feature chains were allowed).

It remains to extend the semantics of $\mathcal{ALC}(\mathcal{D})$ to the new feature forming operator:

Definition 6 (Extended Semantics) An $\mathcal{ALC}(\mathcal{D}_{\mathsf{agg}})$-interpretation $\mathcal{I}$ is an $\mathcal{ALC}(\mathcal{D})$-interpretation which, additionally, satisfies the following conditions. To define the semantics of aggregated features, we introduce the mapping $M_a^{(R \circ f)^{\mathcal{I}}} : \mathsf{dom}(\mathcal{D}) \to \mathbb{N} \cup \{\infty\}$:

$$M_a^{(R \circ f)^{\mathcal{I}}}(z) := \#\{b \in \Delta^{\mathcal{I}} \mid (a, b) \in R^{\mathcal{I}} \text{ and } f^{\mathcal{I}}(b) = z\}.$$

Thus, $M_a^{(R \circ f)^{\mathcal{I}}}$ is a multiset iff $M_a^{(R \circ f)^{\mathcal{I}}}(z) \in \mathbb{N}$ for all $z \in \mathsf{dom}(\mathcal{D})$, i.e., the cardinalities of the considered sets are always finite. Finally, the semantics of aggregated features is defined as follows:

$$(f_1 \dots f_n \circ \Sigma(R \circ f))^{\mathcal{I}}(a) := \begin{cases} \Sigma^{\mathcal{D}}(M_{a'}^{(R \circ f)^{\mathcal{I}}}) & \text{if } (f_1 \dots f_n)^{\mathcal{I}}(a) = a' \in \Delta^{\mathcal{I}} \text{ and } M_{a'}^{(R \circ f)^{\mathcal{I}}} \text{ is a multiset} \\ \text{undefined} & \text{otherwise,} \end{cases}$$

and $\Sigma^{\mathcal{D}}(M_{a'}^{(R \circ f)^{\mathcal{I}}})$ is called the $(f_1 \dots f_n \circ \Sigma(R \circ f))$-successor of a, provided that it is defined.

We point out two consequences of this definition, which might not be obvious at first sight:

(a) If $(R \circ f)^{\mathcal{I}}(a)$ contains individuals in $\Delta^{\mathcal{I}}$, then these individuals have no influence on $M_a^{(R \circ f)^{\mathcal{I}}}$: it is defined in such a way that it takes only into account $(R \circ f)^{\mathcal{I}}$-successors of a in the concrete domain $\mathsf{dom}(\mathcal{D})$.

(b) There are two reasons for $(\Sigma(R \circ f))^{\mathcal{I}}(a)$ to be not defined. On the one hand, $M_a^{(R \circ f)^{\mathcal{I}}}$ need not be a multiset due to the existence of infinitely many R-successors of a that coincide on their f-successors. On the other hand, the aggregation function Σ may be undefined on $M_a^{(R \circ f)^{\mathcal{I}}}$. For example, the sum of infinitely many positive integers and the minimum of an infinitely descending chain of integers are not defined.

4 Decidability results for $\mathcal{ALC}(\mathcal{D}_{\mathsf{agg}})$

In this section, we will give a first, generic decidability result. It is the only one for $\mathcal{ALC}(\mathcal{D}_{\mathsf{agg}})$, and it is obtained by restricting the set of aggregation functions to min and max. However, the result is rather general with respect to the remainder of the concrete domain—besides being admissible, we only require that the (partial) ordering with respect to which $\mathsf{min}, \mathsf{max}$ are defined is available as a predicate symbol in $\mathsf{pred}(\mathcal{D})$.

Theorem 7 If

- $\mathcal{D}$ is admissible,
- $\mathsf{pred}(\mathcal{D})$ contains a binary relation symbol $P_=$ for equality in $\mathcal{D}$, and a binary relation symbol $P_\leq$ for a partial ordering on $\mathsf{dom}(\mathcal{D})$, and
- $\mathsf{agg}(\mathcal{D}) = \{\mathsf{min}, \mathsf{max}\}$,

then satisfiability and subsumption of $\mathcal{ALC}(\mathcal{D}_{\mathsf{agg}})$-concepts is decidable.

Remark: We suppose that min, max have the standard semantics, that is, for multisets $X \subseteq \mathsf{dom}(\mathcal{D})$ we have

$$\mathsf{min}(X) = \begin{cases} x & \text{if } x \in X \text{ such that, for all } y \in X,\ x \leq^{\mathcal{D}} y, \\ \text{undefined} & \text{otherwise.} \end{cases}$$

$$\mathsf{max}(X) = \begin{cases} x & \text{if } x \in X \text{ such that, for all } y \in X,\ y \leq^{\mathcal{D}} x, \\ \text{undefined} & \text{otherwise.} \end{cases}$$

Proof: In the following, a concrete domain as described in the preconditions of Theorem 7 is called $\mathcal{D}^{\mathsf{max}}_{\mathsf{min}}$.

Using the same technique to get rid of aggregated features as in Section 5.1, we could modify the tableau based algorithm in [Baader & Hanschke,1991] to proof Theorem 7.

Fortunately, there is a shorter way to prove Theorem 7, namely by a reduction to $\mathcal{ALCFP}(\mathcal{D})$[Hanschke,1992]. More precisely, each $\mathcal{ALC}(\mathcal{D}^{\mathsf{max}}_{\mathsf{min}})$-concept D can be translated into an $\mathcal{ALCFP}(\mathcal{D})$-concept $\phi(D)$ such that D is satisfiable iff $\phi(D)$ is satisfiable. In [Hanschke,1992], satisfiability of $\mathcal{ALCFP}(\mathcal{D})$-concepts was shown to be decidable, hence satisfiability of $\mathcal{ALC}(\mathcal{D}^{\mathsf{max}}_{\mathsf{min}})$-concepts is also decidable. Morever, $\mathcal{ALC}(\mathcal{D}^{\mathsf{max}}_{\mathsf{min}})$ is closed under negation, hence subsumption $C \sqsubseteq D$ can be reduced to unsatisfiability of $C \sqcap \neg D$.

We start by introducing $\mathcal{ALCP}(\mathcal{D})$, the fragment of $\mathcal{ALCFP}(\mathcal{D})$ which is required for the translation ϕ.

Definition 8 A chain $u = R_1 \dots R_m$ is called *role/feature chain* if each R_i is either a role or a feature name. For a role/feature chain u and $x, y \in \Delta^{\mathcal{I}} \cup \mathsf{dom}(\mathcal{D})$, we have $(x, y) \in u^{\mathcal{I}}$ iff there are $x_1, \dots, x_{m-1}$ with

$$(x, x_1) \in R_1^{\mathcal{I}}, (x_{m-1}, y) \in R_m^{\mathcal{I}}, \text{ and } (x_i, x_{i+1}) \in R_{i+1}^{\mathcal{I}} \text{ for all } 1 \leq i \leq m-2,$$

where, for a feature name f, we use $(w, z) \in f^{\mathcal{I}}$ for $f^{\mathcal{I}}(w) = z$.

$\mathcal{ALCP}(\mathcal{D})$ is obtained from $\mathcal{ALC}(\mathcal{D})$ by allowing, additionally, for concepts of the form

$$\begin{array}{ll} \forall u_1, \dots, u_n.P & \text{(generalised value restriction),} \\ \exists u_1, \dots, u_n.P & \text{(generalised exists-in restriction).} \end{array}$$

where P is a concrete predicate of arity n and $u_1, \dots, u_n$ are role/feature chains.

An $\mathcal{ALCP}(\mathcal{D})$ interpretation must satisfy, additionally,

$$\begin{array}{rcll} (\forall u_1, \dots, u_n.P)^{\mathcal{I}} & = & \{x \in \Delta^{\mathcal{I}} \mid & \text{For all } y_1, \dots, y_n \text{ with } (x, y_i) \in u_i^{\mathcal{I}} \text{ for all} \\ & & & 1 \leq i \leq n \text{ we have } (y_1, \dots, y_n) \in P^{\mathcal{D}}\} \\ (\exists u_1, \dots, u_n.P)^{\mathcal{I}} & = & \{x \in \Delta^{\mathcal{I}} \mid & \text{There are } y_1, \dots, y_n \text{ with } (x, y_i) \in u_i^{\mathcal{I}} \\ & & & \text{for all } 1 \leq i \leq n \text{ and } (y_1, \dots, y_n) \in P^{\mathcal{D}}\} \end{array}$$

For pure feature chains $u_1, \dots, u_n$, the concept $\exists u_1, \dots, u_n.P$ is, by definition, equivalent to $P(u_1, \dots, u_n)$.

The idea of the translation is to introduce new feature names $f_{\mathsf{min}(R \circ f)}$ and $f_{\mathsf{max}(R \circ f)}$ and to use the new generalised restrictions to make sure that $f_{\mathsf{min}(R \circ f)}(x)$ coincides with the minimum of x's $R \circ f$-fillers.

The translation ϕ from $\mathcal{ALC}(\mathcal{D}^{\mathsf{max}}_{\mathsf{min}})$ to $\mathcal{ALCP}(\mathcal{D})$ is defined inductively on the structure on concepts and trivial for all concept forming operators; the only changes it makes are for aggregated features: Whenever features of the form $f_1 \dots f_k \mathsf{min}(R \circ f)$ (resp. $f_1 \dots f_k \mathsf{max}(R \circ f)$) occur, new feature names $f_{\mathsf{min}(R \circ f)}$ (resp. $f_{\mathsf{max}(R \circ f)}$) are introduced. Then these aggregated features are replaced by feature chains of the form $f_1 \dots f_k f_{\mathsf{min}(R \circ f)}$ (resp. $f_1 \dots f_k f_{\mathsf{max}(R \circ f)}$). Finally, we make sure that the $f_1 \dots f_k f_{\mathsf{min}(R \circ f)}$-filler is the minimum of all $f_1 \dots f_k Rf$-fillers. For this, we add concepts of the form

$$\exists f_1 \dots f_k Rf, f_1 \dots f_k f_{\mathsf{min}(R \circ f)}.P_{=} \sqcap \forall f_1 \dots f_k f_{\mathsf{min}(R \circ f)}, f_1 \dots f_k Rf.P_{\leq}.$$

The first conjunct makes sure that the $f_1 \ldots f_k f_{\mathsf{min}(R \circ f)}$-filler (exists and) coincides with one of the $f_1 \ldots f_k Rf$ -fillers. The second conjunct ensures that the $f_1 \ldots f_k f_{\mathsf{min}(R \circ f)}$-filler is smaller or equal than each $f_1 \ldots f_k Rf$-filler. For max, we add similar concepts. More precisely, ϕ is defined as follows:

$$
\begin{array}{lcl@{\quad}lcl}
\phi(C \sqcap D) & = & \phi C \sqcap \phi(D), & \phi(C \sqcup D) & = & \phi C \sqcup \phi(D) \\
\phi(\exists R.C) & = & \exists R.\phi(C), & \phi(\forall R.C) & = & \forall R.\phi(C) \\
\phi(P(u_1, \ldots, u_n)) & = & \exists \phi(u_1), \ldots, \phi(u_n).P \sqcap \prod\limits_{1 \leq i \leq n} \psi(u_i), & & &
\end{array}
$$

where, for a concrete feature u and $\sum \in \{\mathsf{min}, \mathsf{max}\}$

$$
\phi(u) = \begin{cases} u & \text{if } u \text{ is a feature chain} \\ f_1 \ldots f_k f_{\sum(R \circ f)} & \text{if } u = f_1 \ldots f_k \sum(R \circ f) \end{cases}
$$

$$
\psi(u) = \begin{cases} \top & \text{if } u \text{ is a feature chain} \\ \exists f_1 \ldots f_k Rf, f_1 \ldots f_k f_{\mathsf{max}(R \circ f)}.P_{=} \sqcap \\ \forall f_1 \ldots f_k Rf, f_1 \ldots f_k f_{\mathsf{max}(R \circ f)}.P_{\leq} & \text{if } u = f_1 \ldots f_k \mathsf{max}(R \circ f) \\ \exists f_1 \ldots f_k Rf, f_1 \ldots f_k f_{\mathsf{min}(R \circ f)}.P_{=} \sqcap \\ \forall f_1 \ldots f_k f_{\mathsf{min}(R \circ f)}, f_1 \ldots f_k Rf.P_{\leq} & \text{if } u = f_1 \ldots f_k \mathsf{min}(R \circ f) \end{cases}
$$

By construction, each model of an $\mathcal{ALC}(\mathcal{D}^{\mathsf{max}}_{\mathsf{min}})$-concept D can be transformed into a model of $\phi(D)$ by setting $f^{\mathcal{I}}_{\sum(R \circ f)}(x) := \sum(R \circ f)^{\mathcal{I}}(x)$ for $\sum \in \{\mathsf{min}, \mathsf{max}\}$. Vice versa, each model $\mathcal{I}$ of $\phi(D)$ is also a model of D. ∎

Intuitively, the reason for decidability of $\mathcal{ALC}(\mathcal{D}^{\mathsf{max}}_{\mathsf{min}})$ can be seen in the fact that $\mathsf{min}, \mathsf{max}$ only depend on the "boundaries" of a multiset and not on its "inside"—in contrast to all other standard aggregation functions such as sum or $\mathsf{c}ount$.

5 Decidability results for $\mathcal{CQ}(\mathcal{D}_{\mathsf{agg}})$

In this Section, generic decidability results are presented for concrete domains with aggregation functions different from $\{min, max\}$. In [Baader & Sattler,1997], it is shown that satisfiability and subsumption of $\mathcal{ALC}(\mathcal{D}_{\mathsf{agg}})$-concepts is undecidable for a concrete domain which *contains* the nonnegative integers, provides predicates testing for (binary) equality and for (unary)

equality with 1, and where $\mathsf{agg}(\mathcal{D}) = \{\mathsf{min}, \mathsf{max}, \mathsf{sum}\}$. The undecidability has two sources: The universal value restriction in interaction with aggregation functions and the aggregation function sum. In this section, two decidability results are given. In order to proof these results, we disallow universal value restrictions and restrict the use of negation to concept names. It can then be shown that decidability of the satisfiability problem only depends on the decidability of finite conjunctions of concrete predicates which possibly involve aggregated multiset variables. The second result is more general than the first one, but for the first one (where aggregation functions are restricted to so-called multiple-invariant ones), more efficient reason techniques can be applied.

5.1 A completion algorithm for $\mathcal{CQ}(\mathcal{D}_{\mathsf{agg}})$

In this Section, generic decidable extensions by aggregation $\mathcal{CQ}(\mathcal{D}_{\mathsf{agg}})$ will be presented. In order to yield decidability, the use of universal value restriction is disallowed. As universal value restrictions would come in as negation of existential value restrictions, the use of negation is restricted to concept names.

Definition 9 (Syntax) $\mathcal{CQ}(\mathcal{D}_{\mathsf{agg}})$ denotes the Description Logic that is obtained from $\mathcal{ALC}(\mathcal{D}_{\mathsf{agg}})$ by disallowing universal value restrictions $(\forall R.C)$ and by restricting the use of negation to concept names.

We first present the decision procedure for satisfiability of $\mathcal{CQ}(\mathcal{D}_{\mathsf{agg}})$-concepts where $\mathcal{D}$ is restricted to those aggregation functions that do not depend on the multiplicity of elements in the input multiset, so-called *multiple-invariant* aggregation functions. This makes the decision algorithm easier to understand and allows for a more efficient treatment of existential value restriction. In the next section, we will present a more general decision algorithm that can also handle aggregation functions which are not multiple-invariant, such as count.

Definition 10 An aggregation function $\Sigma \in \mathsf{agg}(\mathcal{D})$ is *multiple-invariant* iff $\Sigma(M)$ only depends on the elements occurring in M and not on their multiplicity. That is, Σ is multiple-invariant iff $\Sigma(M) = \Sigma(M')$ holds for

all multisets $M, M' \in \mathsf{MS}(\mathsf{dom}(\mathcal{D}))$ where, for all $z \in \mathsf{dom}(\mathcal{D})$, we have $M(z) \geq 1$ iff $M'(z) \geq 1$.

Satisfiability of $\mathcal{CQ}(\mathcal{D}_{\mathsf{agg}})$-concepts can be decided by a tableau-based algorithm which tries to construct, for an input concept C_0, a model. To this purpose, it breaks down C_0 into subconcepts, hereby making explicit all constraints on individuals in this model. The attempt to construct this model either fails with obvious inconsistencies—in which case C_0 is unsatisfiable—or it succeeds and ends with a precise description of a model of C_0.

In contrast to the algorithm in [Baader & Hanschke,1991] for $\mathcal{ALC}(\mathcal{D})$, constraints will now also involve variables for multisets over the concrete domain—besides individual variables for elements in the concrete domain. To capture the relation between individual and multiset variables, new constraints will be introduced to make explicit the fact that an individual variable stands for an element of a multiset. Finally, besides concrete individual variables, aggregated multiset variables can occur in predicate restrictions.

Definition 11 (Constraint Systems) Let $\tau = \tau_A \cup \tau_{\mathcal{D}} = \{a, b, c, \ldots\} \cup \{x, y, z, \ldots\}$ be an infinite set of abstract and concrete individual variables, and let $\sigma = \{X, Y, Z, \ldots\}$ be an infinite set of multiset variables. The set of aggregated variables, $\{\Sigma(X) \mid \Sigma \in \mathsf{agg}(\mathcal{D}) \text{ and } X \in \sigma\}$, will be denoted by $\mathsf{agg}(\sigma)$. Constraints are of the form:

$$\begin{aligned}
a : C &\text{ for } a \in \tau_A, C \text{ an } \mathcal{CQ}(\mathcal{D}_{\mathsf{agg}})\text{-concept},\\
(a, b) : R &\text{ for } a, b \in \tau_A, R \in N_R,\\
(a, \ell) : f &\text{ for } a \in \tau_A, \ell \in \tau, f \in N_F,\\
(a, Y) : (R \circ f) &\text{ for } a \in \tau_A, R \in N_R, f \in N_F, Y \in \sigma,\\
P(\alpha_1, \ldots, \alpha_n) &\text{ for } \alpha_i \in \tau_{\mathcal{D}} \cup \mathsf{agg}(\sigma), \text{ and}\\
x : Y &\text{ for } x \in \tau_{\mathcal{D}}, Y \in \sigma.
\end{aligned}$$

Constraints of the form $P(\alpha_1, \ldots, \alpha_n)$ or $x : Y$ are called $\mathcal{D}$-constraints. A *constraint system* is a set of constraints. A variable ℓ is said to be an R-successor (resp. an $f_1 \ldots f_n$-successor) of a in a constraint system S iff $(a, \ell) : R \in S$ (resp. $(a, y_1) : f_1, (y_1, y_2) : f_2, \ldots, (y_{n-1}, \ell) : f_n \in S$). An aggregated variable $\Sigma(Y)$ is said to be an $f_1 \ldots f_n \circ \Sigma(R \circ f)$-successor of a in S iff there is an $f_1 \ldots f_n$-successor b of a in S and $(b, Y) : (R \circ f) \in S$.

The semantics for $\mathcal{CQ}(\mathcal{D}_{\mathsf{agg}})$-concepts is extended to constraints as follows.

Definition 12 (Semantics of constraints) We consider interpretations $\mathcal{I}$ that additionally map individual variables to individuals of the concrete or the abstract domain, and multiset variables to multisets over the concrete domain, i.e.,

$$\begin{array}{rl} a^{\mathcal{I}} \in \Delta^{\mathcal{I}} & \text{for } a \in \tau_A, \\ x^{\mathcal{I}} \in \mathsf{dom}(\mathcal{D}) & \text{for } x \in \tau_{\mathcal{D}}, \\ X^{\mathcal{I}} \in \mathsf{MS}(\mathsf{dom}(\mathcal{D})) & \text{for } X \in \sigma. \end{array}$$

An interpretation $\mathcal{I}$ satisfies a constraint of the form

$$\begin{array}{rl} a:C & \text{iff } x^{\mathcal{I}} \in C^{\mathcal{I}}, \\ (a,b):R & \text{iff } (a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}, \\ (a,\ell):f & \text{iff } f^{\mathcal{I}}(a^{\mathcal{I}}) = \ell^{\mathcal{I}}, \\ (a,Y):(R \circ f) & \text{iff } M_{a^{\mathcal{I}}}^{(R \circ f)^{\mathcal{I}}} = Y^{\mathcal{I}}, \\ P(\alpha_1,\ldots,\alpha_n) & \text{iff } P^{\mathcal{D}}(\alpha_1^{\mathcal{I}},\ldots,\alpha_n^{\mathcal{I}}), \\ x:Y & \text{iff } x^{\mathcal{I}} \in Y^{\mathcal{I}}, \end{array}$$

where for $\alpha_i = \Sigma(X)$ we have $\Sigma(X)^{\mathcal{I}} := \Sigma^{\mathcal{D}}(X^{\mathcal{I}})$.

A constraint system S is *satisfiable* iff there exists an interpretation satisfying all constraints in S. Such an interpretation is called a *model* of S. A constraint system S is $\mathcal{D}$-*consistent* iff the conjunction $S_{\mathcal{D}}$ is satisfiable in $\mathcal{D}$, where

$$S_{\mathcal{D}} := \bigwedge_{P(\alpha_1,\ldots,\alpha_n) \in S} P(\alpha_1,\ldots,\alpha_n) \wedge \bigwedge_{Y \text{ occurs in } S} \{\{x_i \mid x_i : Y \in S\}\} \subseteq Y,$$

$x \in \tau_{\mathcal{D}}$ are variables for elements in $\mathsf{dom}(\mathcal{D})$, $Y \in \sigma$ are variables for multisets over $\mathsf{dom}(\mathcal{D})$, and inclusion is interpreted as multiset inclusion. A constraint system S contains a *clash* iff

- $\{a:C, a:\neg C\} \subseteq S$ for some concept C, or
- $\{(a,x):f, (a,b):f\} \subseteq S$ for a concrete variable $x \in \tau_{\mathcal{D}}$ and an abstract variable $b \in \tau_A$.

A constraint system S contains a *fork* iff for $a \in \tau_A$ and a feature name $f \in N_F$ we have

- $\{(a,\ell):f,(a,\ell'):f\} \in S$ for two distinct variables $\ell,\ell' \in \tau_A$ or $\ell,\ell' \in \tau_{\mathcal{D}}$, or
- $\{(a,Y):(R\circ f),(a,Z):(R\circ f)\} \in S$ for two distinct variables $Y,Z \in \sigma$.

If a constraint system S contains a fork $\{(x,\ell):f,(x,\ell'):f\}$ (resp. $\{(a,Y):(R\circ f),(a,Z):(R\circ f)\}$), then we say that S' is obtained by *fork elimination* from S if S' is obtained from S by replacing each occurrence of ℓ by ℓ' (resp. Y by Z).

The tableau-based *completion algorithm* for deciding satisfiability of $\mathcal{CQ}(\mathcal{D}_{\mathsf{agg}})$-concepts applies the *completion rules* introduced in Figure 2 to constraint systems. The completion algorithm works on a tree where each node is labelled with a constraint system. It starts with the tree consisting of a single leaf, the root, labelled with $S = \{x_0 : C_0\}$, where C_0 is the $\mathcal{CQ}(\mathcal{D}_{\mathsf{agg}})$-concept to be tested for satisfiability. A rule can only be applied to a leaf labelled with a clash-free constraint system. Applying a rule $S \to S_i$, for $1 \leq i \leq n$, to such a leaf leads to the creation of n new successors of this node, each labelled with one of the constraint systems S_i. The algorithm terminates if none of the rules can be applied to any of the leaves.

A constraint system S is *complete* if none of the completion rules can be applied to S. The completion algorithm answers "C is satisfiable" iff after its termination one of the leaves is labelled with a complete, clash-free, and $\mathcal{D}$-consistent constraint system.

Lemma 13 Let C_0 be a $\mathcal{CQ}(\mathcal{D}_{\mathsf{agg}})$-concept involving only multiple-invariant aggregation functions, and let S be a constraint system obtained by applying the completion rules to $\{x_0 : C_0\}$. Then

1. for each completion rule $\mathcal{R}$ that can be applied to S, and for each interpretation $\mathcal{I}$, (i) and (ii) are equivalent.
   (i) $\mathcal{I}$ is a model of S.
   (ii) $\mathcal{I}$ is a model of one of the systems S_i obtained by applying $\mathcal{R}$.
2. if S is a complete, $\mathcal{D}$-consistent, and clash-free constraint system, then S has a model.

1. Conjunction: If $a:(C_1 \sqcap C_2) \in S$ and $a:C_1 \notin S$ or $a:C_2 \notin S$, then

$S \to S \cup \{a:C_1, a:C_2\}$.

2. Disjunction: if $a:(C_1 \sqcup C_2) \in S$ and $a:C_1 \notin S$ and $a:C_2 \notin S$, then

$S \to S_1 = S \cup \{a:C_1\}$,

$S \to S_2 = S \cup \{a:C_2\}$.

3. Existential restriction: If $a:(\exists R.C) \in S$ for a role or feature name R and if there is no R-successor b of a with $b:C \in S$, then

$S \to S \cup \{(a,b):R, b:C\}$,

for a new variable $b \in \tau_A$.

If forks were created, then eliminate these forks.

4. Concrete predicates: If $a:P(u_1,\ldots,u_n) \in S$ and a does not have u_i-successors α_i with $P(\alpha_1,\ldots,\alpha_n) \in S$, then, for each u_i let

$$S_i := \begin{cases} \{(a,b_{i1}):f_{i1}, (b_{i1},b_{i2}):f_{i2}, \ldots, (b_{im_i-1}, y_{im_i}):f_{im_i}\} \\ \qquad \text{if } u_i = f_{i1}f_{i2}\ldots f_{im_i} \\ \{(a,b_{i1}):f_{i1}, (b_{i1},b_{i2}):f_{i2}, \ldots, (b_{im_i-1}, b_{im_i}):f_{im_i}, (b_{im_i}, Y_i):(R_i \circ f_i)\} \\ \qquad \text{if } u_i = f_{i1}f_{i2}\ldots f_{im_i} \circ \Sigma_i (R_i \circ f_i) \end{cases}$$

for new variables $b_{ij} \in \tau_A, y_{im_i} \in \tau_{\mathcal{D}}$, $Y_i \in \sigma$. Let α_i be the u_i-successor of a in S_i. Then

$S \to S \cup \{P(\alpha_1,\ldots,\alpha_n)\} \cup \bigcup_{1 \le i \le n} S_i$.

If forks were created, then eliminate these forks.

5. Element assertions: If $\{(a,b):R, (b,z):f, (a,Y):(R \circ f)\} \subseteq S$ for $z \in \tau_{\mathcal{D}}$ and $z:Y \notin S$ then

$S \to S \cup \{z:Y\}$.

Figure 2: The completion rules for $\mathcal{CQ}(\mathcal{D}_{\mathsf{agg}})$.

3. if S contains a clash or is not $\mathcal{D}$-consistent, then S does not have a model.

4. the completion algorithm terminates when applied to $\{x_0:C_0\}$.

Proof of Lemma 13.1: (ii)⇒(i) is obvious because each S_i obtained by applying the completion rules to S is a superset of S where variables were possibly renamed due to fork elimination.

(i)⇒(ii): We only consider Rules 4 and 5 because Rules 1,2, and 3 are obvious and similar to those used in other tableau-based algorithms.

Let $\mathcal{I}$ be a model of S as defined in the precondition of Rule 4, and let S' be obtained by applying Rule 4 to S. Then $a:P(u_1,\ldots,u_n) \in S$ and for each u_i with $1 \leq i \leq n$, if

- u_i is a feature chain $f_{i1}f_{i2}\ldots f_{im_i}$, then $a^{\mathcal{I}}$ has $f_{i1}\ldots f_{ij}$-successors $c_{ij} \in \Delta^{\mathcal{I}}$ for $1 \leq j \leq m_i - 1$, and an $f_{i1}f_{i2}\ldots f_{im_i}$-successor $z_{im_i} \in \mathsf{dom}(\mathcal{D})$. If we define $b_{ij}^{\mathcal{I}} = c_{ij}$ and $y_{im_i}^{\mathcal{I}} = z_{im_i}$, then $\mathcal{I}$ satisfies S_i as defined in Rule 4.
- u_i is an aggregated feature $f_{i1}f_{i2}\ldots f_{im_i} \circ \sum_i(R_i \circ f_i)$, then $a^{\mathcal{I}}$ has $f_{i1}\ldots f_{ij}$-successors $c_{ij} \in \Delta^{\mathcal{I}}$ for $1 \leq j \leq m_i$. If we define $b_{ij}^{\mathcal{I}} = c_{ij}$ and $Y^{\mathcal{I}} = M_{c_{im_i}}^{(R_i \circ f_i)^{\mathcal{I}}}$, then $Y^{\mathcal{I}}$ is by definition the appropriate multiset, and $\mathcal{I}$ satisfies S_i as defined in Rule 4.

Given $\mathcal{I}$ as extended above to the new variables introduced and α_i as defined in Rule 4, we have that α_i is indeed interpreted as the u_i-successor of a, namely $u_i^{\mathcal{I}}(a^{\mathcal{I}}) = \alpha_i^{\mathcal{I}}$ for all $1 \leq i \leq n$. Since $\mathcal{I}$ satisfies $a:P(u_1,\ldots,u_n)$, we thus have that $\mathcal{I}$ satisfies $P(\alpha_1,\ldots,\alpha_n)$.

Let $\mathcal{I}$ be a model of S, and let S' be obtained by applying Rule 5 to S. Then $\{(a,b):R,(b,z):f,(a,Y):(R \circ f)\} \subseteq S$ and $z \in \tau_{\mathcal{D}}$. Thus $z^{\mathcal{I}}$ is an $R \circ f$-successor of $a^{\mathcal{I}}$ in $\mathsf{dom}(\mathcal{D})$. By definition, $z^{\mathcal{I}} \in M_{a^{\mathcal{I}}}^{(R \circ f)^{\mathcal{I}}}$, and since $\mathcal{I}$ is a model of S, $Y^{\mathcal{I}} = M_{a^{\mathcal{I}}}^{(R \circ f)^{\mathcal{I}}}$. Summing up, $z^{\mathcal{I}} \in Y^{\mathcal{I}}$, and thus $\mathcal{I}$ satisfies $S \cup \{z:Y\} = S'$. ∎

Proof of Lemma 13.2: Let S be a complete, $\mathcal{D}$-consistent, and clash-free constraint system involving concrete and multiset variables $\{x_1,\ldots,x_m, X_1,\ldots, X_n\}$, and let $\{\{\hat{x}_1,\ldots,\hat{x}_m,\hat{X}_1,\ldots,\hat{X}_n\}\}$ be a solution for the conjunction $S_{\mathcal{D}}$. Hence we have $\{\{\hat{x}_j \mid x_j:X_i \in S\}\} \subseteq \hat{X}_i$ for all multiset variables X_i

occurring in S. We define $\mathcal{I}'$ as follows:

$$
\begin{array}{rcl}
\Delta^{\mathcal{I}'} & := & \tau_A, \\
a^{\mathcal{I}'} & := & a \text{ for abstract variables } a \in \tau_A, \\
x^{\mathcal{I}'} & := & \hat{x} \text{ for concrete variables } x \in \tau_{\mathcal{D}}, \\
X^{\mathcal{I}'} & := & \hat{X} \text{ for multiset variables } X \in \sigma, \\
A^{\mathcal{I}'} & := & \{b \in \Delta^{\mathcal{I}'} \mid b : A \in S\} \text{ for concept names } A \in N_C, \\
R^{\mathcal{I}'} & := & \{(a,b) \in \Delta^{\mathcal{I}'} \times \Delta^{\mathcal{I}'} \mid (a,b) : R \in S\} \text{ for role names } R \in N_R, \\
f^{\mathcal{I}'}(c) & := & \begin{cases} b & \text{if } (c,b) : f \in S \text{ for } b \in \tau_A, \\ \hat{x} & \text{if } (c,x) : f \in S \text{ for } x \in \tau_{\mathcal{D}}, \\ \text{undefined} & \text{else.} \end{cases} \\
 & & \qquad \text{for feature names } f \in N_F.
\end{array}
$$

Feature names f are well-defined because S is clash-free. The only reason why $\mathcal{I}'$ might not be a model of S is the following: An abstract individual a may have less R-successors having an f-successor in $\mathsf{dom}(\mathcal{D})$ than required by the solution for the corresponding multiset variable X_i, that is, for constraints $(a, X_i) : (R \circ f) \in S$ we might have $M_{a^{\mathcal{I}'}}^{(R \circledcirc f)^{\mathcal{I}'}} \subsetneq \hat{X}_i$. Due to the absence of universal value restrictions, a model $\mathcal{I}$ can be obtained from $\mathcal{I}'$ by simply adding R-successors d_a^{Rfj} and the lacking $R \circledcirc f$-successors $\hat{y}_a^{Rfj}$. More precisely, for a multiset variable X_i with $(a, X_i) : (R \circ f) \in S$, let

$$\hat{X}_i \setminus M_a^{(R \circledcirc f)^{\mathcal{I}'}} := \{\{\hat{y}_a^{Rf1}, \ldots, \hat{y}_a^{Rfm_i}\}\}.$$

Then

$$
\begin{array}{rcl}
\Delta^{\mathcal{I}} & := & \Delta^{\mathcal{I}'} \uplus \bigcup\limits_{\substack{1 \le i \le n \\ (a, X_i) : (R \circledcirc f) \in S}} \{d_a^{Rf1}, \ldots, d_a^{Rfm_i}\}, \\
A^{\mathcal{I}} & := & A^{\mathcal{I}'}, \\
R^{\mathcal{I}} & := & R^{\mathcal{I}'} \cup \bigcup\limits_{\substack{1 \le i \le n \\ (a, X_i) : (R \circledcirc f) \in S}} \{(a, d_a^{Rfj}) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid 1 \le j \le m_i\} \\
 & & \text{for role names } R \in N_R, \\
f^{\mathcal{I}}(c) & := & \begin{cases} b & \text{if } (c,b) : f \in S, \\ \hat{x} & \text{if } (c,x) : f \in S, \\ \hat{y}_a^{Rfj} & \text{if } c = d_a^{Rfj} \\ \text{undefined} & \text{else.} \end{cases} \qquad \text{for feature names } f \in N_F
\end{array}
$$

Obviously, $\hat{X}_i = M_{a^{\mathcal{I}}}^{(R \circledcirc f)^{\mathcal{I}}}$ for all multiset variables with $(a, X_i) : (R \circ f) \in S$. Furthermore, $\{\hat{x}_1, \ldots, \hat{x}_m, \hat{X}_1, \ldots, \hat{X}_n\}$ being a solution for $S_{\mathcal{D}}$ implies that

$\mathcal{I}$ satisfies all $\mathcal{D}$-constraints in S. By induction on the structure of concepts, it can be easily shown that $\mathcal{I}$ satisfies all constraints in S.

By definition, $\mathcal{I}$ satisfies all constraints of the form $b:A$ for concept names A. Since S is clash-free, $\mathcal{I}$ satisfies all constraints of the form $b:\neg A$. By induction and because S is complete, $\mathcal{I}$ satisfies all constraints of the form $a:(C_1 \sqcap C_2)$ and $a:(C_1 \sqcup C_2)$. The same arguments imply that $\mathcal{I}$ satisfies constraints of the form $a:(\exists R.C)$ for role or feature names R. ∎

Remark: This extension of $\mathcal{I}'$ to a model $\mathcal{I}$ of a complete and clash-free constraint system was only possible because we disallowed the use of universal value restriction: This enables us to add lacking $R \circ f$-successors without the necessity to check again whether the new, intermediate R-successors satisfy the universal value restrictions.

Proof of Lemma 13.3: If S contains a clash, then S is obviously unsatisfiable. A model $\mathcal{I}$ of S satisfies all $\mathcal{D}$-constraints, hence $\mathcal{I}$ yields a solution for

$$\bigwedge_{P(\alpha_1,\ldots,\alpha_n)\in S} P(\alpha_1,\ldots,\alpha_n) \wedge \bigwedge_{Y \text{ occurs in } S} \{x_i \mid x_i:Y \in S\} \subseteq Y$$

where inclusion is interpreted as set inclusion. This is so because $\mathcal{I}$ satisfies all predicate restrictions and all constraints of the form $x_i:Y$. Hence $\{\{x_i^{\mathcal{I}} \mid x_i:Y \in S\}\}(x) \geq 1$ implies $Y^{\mathcal{I}}(x) \geq 1$ for all $x \in \mathsf{dom}(\mathcal{D})$, and the only reason why $\mathcal{I}$ might not satisfy $S_{\mathcal{D}}$ is $\{\{x_i^{\mathcal{I}} \mid x_i:Y \in S\}\} \not\subseteq Y^{\mathcal{I}}$, which can happen only because there are some elements in $Y^{\mathcal{I}}$ who occur less often in $Y^{\mathcal{I}}$ than in $\{\{x_i^{\mathcal{I}} \mid x_i:Y \in S\}\}$, namely $Y^{\mathcal{I}}(x) < \{\{x_i^{\mathcal{I}} \mid x_i:Y \in S\}\}(x)$ for some $x \in \mathsf{dom}(\mathcal{D})$. Since all aggregation functions are multiple-invariant, the multiplicity of these elements in $Y^{\mathcal{I}}$ can be increased such that $Y^{\mathcal{I}}(x) \geq \{\{x_i^{\mathcal{I}} \mid x_i:Y \in S\}\}(x)$, which yields a solution of $S_{\mathcal{D}}$. Hence each model of S yields of solution for $S_{\mathcal{D}}$. By contraposition, a constraint system that is not $\mathcal{D}$-consistent cannot have a model. ∎

Proof of Lemma 13.4: Termination is an immediate consequence of the following arguments. (a) All concepts of constraints added by the completion rules are subconcepts of the concept C_0. It is not difficult to see that the number of subconcepts of a concept C is linear in the length of C. (b) If a rule $\mathcal{R}$ can be applied to a constraint system S, then this is because of the

presence of a particular constraint $a:C$ in S, and the application of $\mathcal{R}$ adds constraints whose concepts are strictly shorter than C. (c) The introduction of role- or feature successors is always triggered by a constraint of the form $x:\exists R, C$ or $a:P(u_1,\ldots,u_n)$, hence each variable has only finitely many role successors. (d) Once a variable is introduced, it is present in all subsequent constraint systems, and constraints are never removed (these two properties prevent the algorithm from looping).

5.2 Extension to arbitrary aggregation functions

The proof of soundness and completeness of the completion algorithm in the previous section was only possible because all aggregation functions were supposed to be multiple-invariant. Without this property, the completion algorithm presented in the previous section would be incomplete. For example, the concept C below is obviously satisfiable, and each instance of C has exactly one R-successor.

$$C := (\exists R.\geq_2(f)) \sqcap (\exists R.=_2(f)) \sqcap \leq_1(\mathsf{count}(R\circ f))$$

C involves the aggregation function count which is not multiple-invariant. This causes incompleteness of the completion algorithm presented in the previous section: Although C is satisfiable, it would not generate a complete, clash-free and $\mathcal{D}$-consistent constraint system for C, but would end up with a non-$\mathcal{D}$-consistent constraint system: It would generate two R-successors and two $R\circ f$-successors, say x, y, which yields a constraint system that is not $\mathcal{D}$-consistent because $\leq_1 (\mathsf{count}(Y)) \wedge \{\{x, y\}\} \subseteq Y$ is contradictory. To identify these two R-successors (and hereby the two $R\circ f$-successors) is not a good idea because they might have to satisfy contradicting constraints. For example, the concept C' below is unsatisfiable: It requires 2 R-successors which cannot be interpreted by the same abstract individual because they have contradicting restrictions on their g-successors, and it has at most 1 R-successor:

$$C' := (\exists R.\geq_2(f) \sqcap =_3(g)) \sqcap (\exists R.=_2(f) \sqcap =_7(g)) \sqcap \leq_1(\mathsf{count}(R\circ f)).$$

To design a completion algorithm which is also complete for concepts involving not multiple-invariant aggregation functions, Rule 3 in Figure 2 is

replaced by Rules 3.a, 3.b in Figure 3. These rules are designed such that the proof does not depend on multiple-invariance. To this purpose, all possibilities to generate as few R-successors as possible are tested. This is realized by trying to reuse, for each $a:(\exists R.C)$ constraint, already existing R-successors of a. For the case that this is not possible, a new R-successor is also introduced. As a consequence, we can restrict our attention to those models that interprete different R-successors as different individuals.

Definition 14 An *m-model* $\mathcal{I}$ of a $\mathcal{CQ}(\mathcal{D}_{\mathsf{agg}})$-constraint system S is a model that satisfies, additionally, $b^{\mathcal{I}} \neq c^{\mathcal{I}}$ for all $b, c \in \tau_A$ with $\{(a,b):R, (a,b):R\} \subseteq S$ for some $a \in \tau_A$ and $R \in N_R$.

Like in the previous section, we will present a technical lemma that implies that decidability of satisfiability of $\mathcal{CQ}(\mathcal{D}_{\mathsf{agg}})$-concepts only depends on the decidability of $\mathcal{D}$-consistency. Examples for decidable concrete domains which have min, max and count as aggregation functions will be presented below.

Theorem 15 If $\mathcal{D}$ is a concrete domain such that $\mathcal{D}$-consistency is decidable, then satisfiability of $\mathcal{CQ}(\mathcal{D}_{\mathsf{agg}})$-concepts is decidable.

Lemma 16 Let C_0 be an $\mathcal{CQ}(\mathcal{D}_{\mathsf{agg}})$-concept, and let S be a constraint system obtained by applying the modified completion rules to $\{x_0 : C_0\}$. Then

1. if C_0 is satisfiable, then $\{x_0 : C_0\}$ has an m-model.
2. For each completion rule $\mathcal{R}$ that can be applied to S, and for each interpretation $\mathcal{I}$, (i) and (ii) are equivalent.
   (i) $\mathcal{I}$ is an m-model of S.
   (ii) $\mathcal{I}$ is an m-model of one of the systems S_i obtained by applying $\mathcal{R}$.
3. If S is a complete, $\mathcal{D}$-consistent, and clash-free constraint system, then S has an m-model.
4. If S contains a clash or is not $\mathcal{D}$-consistent, then S does not have an m-model.

3.a. Existential restriction on features: If $a:(\exists f.C) \in S$ for a feature name f and if there is an f-successor b of a with $b:C \notin S$, then

$S \quad \rightarrow S \cup \{b:C\}$

otherwise, if a has no f-successor, then

$S \quad \rightarrow S \cup \{(a,b):f, b:C\}$.

for a new variable $b \in \tau_A$.

3.b. Existential restriction on roles: If $a:(\exists R.C) \in S$ for a role name R, $\{b_1,\ldots,b_n\}$ are all R-successors of a, and $b_i:C \notin S$ for all b_i, then

$S_i \quad \rightarrow S \cup \{b_i:C\}$

$S_{n+1} \quad \rightarrow S \cup \{(a,b):R, b:C\}$

for a new variable $b \in \tau_A$.

Figure 3: The modified completion rule for existential restrictions.

5. The completion algorithm terminates when applied to $\{x_0:C_0\}$.

In the following, we will only proof those parts of this lemma that are not identical to the proof of Lemma 13.

Proof of Lemma 16.1 Each model of a constraint system $\{x_0:C_0\}$ is obviously an m-model.

Proof of Lemma 16.2:(ii)⇒(i) Rules 3.a and 3.b:

Let S be as specified in the precondition of Rule 3.a with $a:(\exists f.C)$, and let $\mathcal{I}$ be an m-model of S with $f^{\mathcal{I}}(a^{\mathcal{I}}) = c$ and $c \in C^{\mathcal{I}}$. Hence $c \in \Delta^{\mathcal{I}}$. If there exists a variable $b \in \tau_A$ with $(a,b):f \in S$, then we have $b^{\mathcal{I}} = c$ because $\mathcal{I}$ is a model of S and $\mathcal{I}$ satisfies $S \cup \{b:C\}$. Otherwise, $\mathcal{I}$ satisfies $S \cup \{(a,b):f, b:C\}$ for a new variable b if we define $b^{\mathcal{I}} = c$. In both cases, $\mathcal{I}$ is still an m-model.

Let S be as specified in the precondition of Rule 3.b with $a:(\exists R.C)$, and let $\mathcal{I}$ be an m-model of S with $b_i^{\mathcal{I}} = c_i$ for all R-successors b_i of a. If, for an R-successor b_i of a, we have $b_i^{\mathcal{I}} \in C^{\mathcal{I}}$ then $\mathcal{I}$ is obviously an m-model of $S \cup \{b_i:C\}$. Otherwise, there is some $c \in \Delta^{\mathcal{I}}$ with $(a^{\mathcal{I}},c) \in R^{\mathcal{I}}$, $c \in C^{\mathcal{I}}$

and $c \neq b_i^{\mathcal{I}}$ for all R-successors b_i of a. In this case, $\mathcal{I}$ is an m-model of $S \cup \{(a,b) : R, b : C\}$.

Proof of Lemma 16.3 is identical to the proof of Lemma 13.2 because the model constructed in this proof for a complete, clash-free and $\mathcal{D}$-consistent constraint system is an m-model.

Proof of Lemma 16.4 Similar to the proof of Lemma 13.3, an m-model $\mathcal{I}$ of S yields a solution for

$$\bigwedge_{P(\alpha_1,\ldots,\alpha_n)\in S} P(\alpha_1,\ldots,\alpha_n) \wedge \bigwedge_{Y \text{ occurs in } S} \{x_i \mid x_i : Y \in S\} \subseteq Y$$

where inclusion is interpreted as set inclusion. Now, $\mathcal{I}$ is an m-model of S, hence for each variable a, for each pair of R-successors b_i, b_j of a in S, we have $b_i^{\mathcal{I}} \neq b_j^{\mathcal{I}}$. By definition of the semantics of constraints, we have $X^{\mathcal{I}} = M_{a^{\mathcal{I}}}^{(R \circ f)^{\mathcal{I}}}$ for multiset variables X with $(a, X) : (R \circ f) \in S$. As a consequence, $\{\{x_i^{\mathcal{I}} \mid x_i : Y \in S\}\} \subseteq Y^{\mathcal{I}}$, thus $\mathcal{I}$ yields a solution for $S_{\mathcal{D}}$, and S is $\mathcal{D}$-consistent.

■

5.3 Examples for decidable concrete domains

In the following, we will give examples for concrete domains $\mathcal{D}$ for which $\mathcal{D}$-consistency can be decided.

Lemma 17 For a concrete domain $\mathcal{D}$ where

- $\mathsf{dom}(\mathcal{D})$ is the set of nonnegative integers, integers, rational numbers or reals with the natural order $<$,
- $\mathsf{pred}(d) = \{<, \geq, >, \leq, =, \} \cup \bigcup_{n \in \mathsf{dom}(\mathcal{D})} \{\leq_n, \geq_n, >_n, <_n, =_n\}$ (where $>$ is the inverse of $<$, $\leq, \geq$ are, respectively, the reflexive closures of $<, >$, and $\{=_n, \leq_n, \geq_n, >_n, <_n\}$ are unary predicates testing for equality with n (resp. for being less or equal, greater or equal, greater, and less than n).
- $\mathsf{agg}(\mathcal{D}) = \{\mathsf{min}, \mathsf{max}\}$ with the obvious meaning,

$\mathcal{D}$-consistency of $\mathcal{CQ}(\mathcal{D}_{\mathsf{agg}})$-constraint systems is decidable.

Proof of Lemma 17: Let S be a constraint system, let $\mathcal{D}$ be defined as in the precondition of Lemma 17, and let S_N be the set of $\mathcal{D}$-constraints in S.

$\mathcal{D}$-consistency of S is decided by transforming S_N into a set D_S of (in)equalities without aggregation functions that is satisfiable iff S is $\mathcal{D}$-consistent. Satisfiability of D_S can then be easily decided using, for example, graph-based techniqes. To this purpose, for each term $\mathsf{max}(Y)$ (resp. $\mathsf{min}(Y)$) occurring in S_N, a new variable y_{max} (resp. y_{min}) is introduced in an intermediate set of constraints D'_S. More precisely, D'_S is obtained from S_N by replacing each occurrence of $\mathsf{max}(Y)$ (resp. $\mathsf{min}(Y)$) by y_{max} (resp. y_{min}), i.e.,

$$D'_S := \; S_N[\mathsf{max}(Y)/y_{\mathsf{max}} \text{ for } Y \in \sigma][\mathsf{min}(Y)/y_{\mathsf{min}} \text{ for } Y \in \sigma].$$

Then D_S is obtained from D'_S by replacing constraints by appropriate (in)equalities and adding axioms to capture the interaction between $\mathsf{min}(Y), \mathsf{max}(Y)$ and $z:Y$, i.e.,

$$\begin{aligned} D_S := \; & \{y_{\mathsf{min}} \leq y_{\mathsf{max}} \mid y_{\mathsf{min}} \text{ or } y_{\mathsf{max}} \text{ occurs in } D'_S\} \cup \\ & \{y_{\mathsf{min}} \leq z \mid y_{\mathsf{min}} \text{ occurs in } D'_S \text{ and } (z:Y) \in S_N\} \cup \\ & \{y_{\mathsf{max}} \geq z \mid y_{\mathsf{min}} \text{ occurs in } D'_S \text{ and } (z:Y) \in S_N\} \cup \\ & \{x\, P\, y \mid P \in \{\leq, \geq, >, <, =\} \text{ and } P(x,y) \in D'_S\} \cup \\ & \{x\, P\, n \mid P_n \in \{\leq_n, \geq_n, >_n, <_n, =_n\} \text{ and } P_n(x) \in D'_S\} \end{aligned}$$

Lemma 18 D_S is satisfiable iff S is $\mathcal{D}$-consistent.

Proof: (a) the only constraints involved on $\mathsf{min}(Y), \mathsf{max}(Y)$ are that $\mathsf{min}(Y)$ is less and $\mathsf{max}(Y)$ is greater than each element in Y, and (b) the only elements that are required to be in Y are x_i with $x_i:Y \in S$, the minimum, and the maximum of Y. Each solution of $S_{\mathcal{D}}$ is clearly also a solution of D_S. Now suppose we have a solution of D_S where $\hat{x} \in \mathsf{dom}(\Delta^{\mathcal{I}})$ is the value for each variable x in D_S. Then we can define a solution for $S_{\mathcal{D}}$ by

$$\hat{Y} := \{\{\hat{x} \mid x:Y \in S\}\} \cup \{\{\hat{y}_{\mathsf{min}}, \hat{y}_{\mathsf{max}}\}\}.$$

Since $\mathsf{min}, \mathsf{max}$ are multiple-invariant and we started from a solution of D_S, this solution satisfies all predicate restrictions in S. Furthermore, the solution satisfies $\mathsf{max}(\hat{Y}) = \hat{y}_{\mathsf{max}}$ and $\mathsf{min}(\hat{Y}) = \hat{y}_{\mathsf{min}}$. By definition, this solution also satisfies the multiset inclusion conjuncts in $S_{\mathcal{D}}$. ∎

It remains to show that satisfiability of a set D_S of (in)equalities for $\mathcal{D}$-constraints in constraint systems S is decidable. Since all (in)equalities in D_S are linear, each set D_S of (in)equalities is satisfiable over the real numbers iff it is decidable over the rational numbers. Furthermore, the following transformation obviously preserve satisfiability:

1. Remove equations of the form $x = x$ from D_S.
2. Replace x by y if $x = y \in D_S$ for two variables x, y.
3. Replace x by n if $x = n \in D_S$ for a variable x and a constant $n \in \mathsf{dom}(\mathcal{D})$.
4. If the concrete domain is the set of nonnegative integers, add inequalities $x \geq 0$ for all variables.

Without loss of generality, we thus restrict our attention to sets D_S without equalities, and to solutions in the integers or in the rationals. Finally, we may orient all inequalities such that only $<$ and $\leq$ occur in D_S.

We construct a graph G_S for D_S as follows: G_S's nodes are the constants and variables in D_S. The edges are labelled with $<$ or $\leq$: Two nodes x, y are related via a $<$-edge (resp. $\leq$) iff the corresponding constraint $x \leq y$ (resp. $x < y$) is in D_N, and they are related via a $<$-edge if both nodes are constants in $\mathsf{dom}(\mathcal{D})$ with $x < y$.

Lemma 19 D_S is unsatisfiable over the

1. rational numbers iff G_S has a cycle with at least one $<$-edge.
2. integers iff G_S has a cycle with at least one $<$-edge or if a path from a constant c_1 to a constant $c_2 > c_1$ involve at more than $c_2 - c_1$ $<$-edges.

Proof: The if direction is obvious. Now let G_S be a graph without cycles involving $<$-egdes (and where paths between constants don't involve too much $<$-edges). To obtain a solution for D_S, we identify all $\leq$-cycles and perform topological sorting on the remainder. In the case we are looking for a solution in the integers, we have to take care that the smallest integers

associated to variables not having a lower bound are really small enough. A save starting point could be the smalles constant in G_S minus the number of variables occuring in G_S. To choose rational numbers for variables bound between c_1 and C_2, we choose $c_1 + 1/2(c_2 - c_1)$ for the smallest one, say x_1, and $x_1 + 1/2(c_2 - x_1)$ for the next one, etc. ∎

Lemma 20 For a concrete domain $\mathcal{D}$ where

- $\mathsf{dom}(\mathcal{D})$ is the set of nonnegative integers, integers, real or rational numbers,
- $\mathsf{pred}(d) = \{<, \geq, >, \leq, =, \} \cup \bigcup_{n \in \mathsf{dom}(\mathcal{D})} \{\leq_n, \geq_n, >_n, <_n, =_n\}$ (where $>$ is the inverse of $<$ and $\{=_n, \leq_n, \geq_n, >_n, <_n\}$ are unary predicates testing for equality with n (resp. for being less or equal, greater or equal, greater, and less than n).
- $\mathsf{agg}(\mathcal{D}) = \{\mathsf{min}, \mathsf{max}, \mathsf{count}\}$ with the obvious meaning,

$\mathcal{D}$-consistency of $\mathcal{CQ}(\mathcal{D}_{\mathsf{agg}})$-constraint systems is decidable.

Proof of Lemma 20: The decision procedure is similar to the one given in the proof of Lemma 17, with the only difference that, in addition, aggregated multiset variables involving count are also replaced by appropriate individual variables y_{count}, and that the behaviour of count has to be axiomatised. More precisely, given a constraint system S for $\mathcal{CQ}(\mathcal{D}_{\mathsf{agg}})$-concepts for a concrete domain $\mathcal{D}_{\mathsf{agg}}$ as described in Lemma 20, the sets D_S, D'_S are defined as follows:

$$
\begin{array}{rl}
D'_S := & S_N[\mathsf{max}(Y)/y_{\mathsf{max}} \text{ for } Y \in \sigma][\mathsf{min}(Y)/y_{\mathsf{min}} \text{ for } Y \in \sigma] \\
& \quad [\mathsf{count}(Y)/y_{\mathsf{count}} \text{ for } Y \in \sigma] \\
D_S := & \{D_{y_{\mathsf{count}}} \mid y_{\mathsf{count}} \text{ occurs in } D'_S\} \cup \\
& \{y_{\mathsf{min}} \leq y_{\mathsf{max}} \mid y_{\mathsf{min}} \text{ or } y_{\mathsf{max}} \text{ occurs in } D'_S\} \cup \\
& \{y_{\mathsf{min}} \leq z \mid y_{\mathsf{min}} \text{ occurs in } D'_S \text{ and } (z:Y) \in S_N\} \cup \\
& \{y_{\mathsf{max}} \geq z \mid y_{\mathsf{min}} \text{ occurs in } D'_S \text{ and } (z:Y) \in S_N\} \cup \\
& \{x\, P\, y \mid P \in \{\leq, \geq, >, <, =\} \text{ and } P(x,y) \in D'_S\} \cup \\
& \{x\, P\, n \mid P_n \in \{\leq_n, \geq_n, >_n, <_n, =_n\} \text{ and } P_n(x) \in D'_S\}
\end{array}
$$

where the formula $D_{y_{\mathsf{count}}}$ is defined as follows. For a better readability, we use $x_Y := \{x \in \tau_{\mathcal{D}} \mid x : Y \in S\}$ as a shorthand for those concrete variables belonging to Y.

$$\begin{aligned} D_{y_{\mathsf{count}}} := \; & \Big(\big(\#x_Y = y_{\mathsf{count}} \wedge \textstyle\bigvee_{x \in x_Y} x = y_{\mathsf{min}} \wedge \bigvee_{x \in x_Y} x = y_{\mathsf{max}}\big) \vee \\ & \big(\#x_Y = y_{\mathsf{count}} + 1 \wedge \textstyle\bigvee_{x \in x_Y} (x = y_{\mathsf{min}} \vee x = y_{\mathsf{max}})\big) \vee \\ & (\#x_Y \geq y_{\mathsf{count}} + 2) \Big) \wedge \\ & y_{\mathsf{count}} \in \mathbb{Z} \wedge y_{\mathsf{count}} \geq 0 \end{aligned}$$

The disjunction is necessary because we have to distinguish between the case where some of the concrete variables known to belong to a multiset coincide with its minimum and/or maximum (in which case the cardinality can be equal to $\#x_Y$, reps. $\#x_Y + 1$), and the case where both the minimum and the maximum are distinct from values for concrete variables in x_Y. This distinction is necessary because constraints such as

$$\{x : Y, =_4 (x), \geq_6 (\mathsf{max}(Y), \leq_2 (\mathsf{min}(Y))\}$$

impose a cardinality less or equal to $\#x_Y + 2$, where $\#x_Y = 1$.

Disjunctions can be removed by transforming $D_{y_{\mathsf{count}}}$ into disjunctive normal form, and testing satisfiability of D_S separately for each disjunct together with the other (in)equalities. Thus we assume that we still have to decide satisfiability of a set of (in)equalities. Similar to the proof of Lemma 20, this problem can be reduced to satisfiability of inequalities in the integers or the rational numbers—with the only difference that variables y_{count} stand always for integers. In the following, let D_S stand for such a set of inequalities involving constants and both variables for integers and rationals.

Again, satisfiability of these inequalities can decided by constructing the graph G_S for D_S in the same way we constructed a graph for D_S, and testing this graph for cycles involving $<$ and paths between 2 nodes representing constants. Due to the presence of y_{count}, each node is now either associated with a constant, an integer variable (either because this node stands for y_{count} or we are looking for a solution in the (nonnegative) integers), or a rational variable (if we are looking for a solution in the rational or real numbers). This mixture makes the description of a path between two constants that involves too many different integers rather complicated, and we will need the following definitions for its formulation.

Let c be a constant in $\mathsf{dom}(\mathcal{D})$. Then $\lfloor c \rfloor$ denotes the largest integer less or equal c and $\lceil c \rceil$ denotes the smallest integer greater or equal c. For an integer variable x, we define the possibly smallest value $\mathsf{lower}(c, x)$ above c for x by

$$\mathsf{lower}(c, x) := \begin{cases} c & \text{if } c \in \mathbb{Z} \text{ and no path from } c \text{ to } x \text{ involves a } <\text{-edge} \\ \lfloor c+1 \rfloor & \text{otherwise.} \end{cases}$$

Similar, the possibly largest value $\mathsf{up}(c, x)$ is defined as follows:

$$\mathsf{up}(c, x) := \begin{cases} c & \text{if } c \in \mathbb{Z} \text{ and no path from } x \text{ to } c \text{ involves a } <\text{-edge} \\ \lceil c-1 \rceil & \text{otherwise.} \end{cases}$$

The last condition in the following Lemma describes the situation where, between two constants c_1, c_2, there is not enough space for a chain of ascending integers.

Lemma 21 *D_S is unsatisfiable over a mixture of integers and rational numbers iff G_S*

- has a cycle with at least one $<$-edge or
- there are constants $c_2 > c_1$, a path from c_1 to c_2 involving integer variables $x_1, \ldots, x_m$, where
  - there is a path involving a $<$-edge from each x_i to x_{i+1}, and
  - $m > \mathsf{up}(c_2, x_m) - \mathsf{lower}(c_1, x_1) + 1$.

As a consequence of the observations made in this section, we have the following decidability result.

Corollary 1 *If $\mathcal{D}$ is a concrete domain such that*

- *$\mathsf{dom}(\mathcal{D})$ is the set of nonnegative integers, integers, real or rational numbers,*
- *$\mathsf{pred}(d) = \{<, \geq, >, \leq, =, \} \cup \bigcup_{n \in \mathsf{dom}(\mathcal{D})} \{\leq_n, \geq_n, >_n, <_n, =_n\}$ (where $>$ is the inverse of $<$ and $\{=_n, \leq_n, \geq_n, >_n, <_n\}$ are unary predicates testing for equality with n (resp. for being less or equal, greater or equal, greater, and less than n).*
- *$\mathsf{agg}(\mathcal{D}) = \{\mathsf{min}, \mathsf{max}, \mathsf{count}\}$ with the obvious meaning,*

then satisfiability of $\mathcal{CQ}(\mathcal{D}_{\mathsf{agg}})$-concepts is decidable.

6 Related and future work

We have presented an expressive Description Logic that can also express properties involving aggregated data. The results of this paper are thus not only of interest for knowledge representation, but also for database research, for example, in the context of intensional reasoning in the presence of aggregation, as considered in [Ross *et al.*,1998; Gupta *et al.*,1995; Mumick & Shmueli,1995; Levy & Mumick,1996; Srivastava *et al.*,1996]. The undecidability results are orthogonal to those presented in [Mumick & Shmueli,1995] since our prerequisites are weaker and no recursion mechanisms are used. Neither are they implied by the undecidability results in [Ross *et al.*,1998]: the results presented there concern aggregation constraints involving addition as well as rather complex aggregation functions like average and count. The decidability results are also orthogonal to the decidability results [Nutt *et al.*,] for containment of conjunctive queries with aggregation in the query head: we have fewer aggregation functions, but allow to use them in a more complex way. The exact connection between our decidable DL $\mathcal{CQ}(\mathcal{D}_{\mathsf{agg}})$ and conjunctive queries with aggregation is a topic for future research.

The decision procedure for $\mathcal{CQ}(\mathcal{D}_{\mathsf{agg}})$ is parameterised by a decision procedure for $\mathcal{D}$-consistency of the concrete domain with aggregation. Thus, it is important to find additional concrete domains with aggregation for which $\mathcal{D}$-consistency is decidable. For example, what happens if the aggregation function count in Lemma 17 is replaced by sum? It should be noted that *adding* sum to the concrete domain considered in the lemma makes $\mathcal{D}$-consistency undecidable. This is as an easy consequence of one of the undecidability result in [Ross *et al.*,1998].

References

[Baader & Hanschke, 1991] F. Baader and P. Hanschke. A schema for integrating concrete domains into concept languages. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (IJCAI-91)*, pages 452–457, Sydney, 1991.

[Baader & Sattler, 1997] F. Baader and U. Sattler. Description logics with aggregates and concrete domains. Technical Report 97-01, LuFg Theoreti-

cal Computer Science, RWTH Aachen, 1997. Available via www: `http://www-lti.informatik.rwth-aachen.de/Forschung/Papers.html`.

[Donini *et al.*, 1991] F. Donini, M. Lenzerini, D. Nardi, and W. Nutt. The complexity of concept languages. In *Proceedings of the Second International Conference on the Principles of Knowledge Representation and Reasoning (KR-91)*, Boston, MA, USA, 1991.

[Donini *et al.*, 1995] F. M. Donini, M. Lenzerini, D. Nardi, and W. Nutt. The complexity of concept languages. Technical Report RR-95-07, Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI), Kaiserslautern, Germany, 1995.

[Gupta *et al.*, 1995] A. Gupta, V. Harinarayan, and D. Quass. Aggregate-query processing in data warehousing environments. In *Proceedings of the 21. International Conference on Very Large Data Bases (VLDB-95)*, 1995.

[Hanschke, 1992] P. Hanschke. Specifying Role Interaction in Concept Languages. In *Proceedings of the Second International Conference on the Principles of Knowledge Representation and Reasoning (KR-91)*, 1992.

[Hollunder *et al.*, 1990] B. Hollunder, W. Nutt, and M. Schmidt-Schauss. Subsumption algorithms for concept description languages. In *ECAI-90*, Pitman Publishing, London, 1990.

[Levy & Mumick, 1996] A. Y. Levy and I. S. Mumick. Reasoning with aggregation constraints. In *Proceedings of the International Conference on Extending Database Technology (EDBT-96)*, Avignon, France, 1996.

[Mumick & Shmueli, 1995] I. S. Mumick and O. Shmueli. How expressive is stratified aggregation. *Annals of Mathematics and Artificial Intelligence*, 15(3-4), 1995.

[Nutt *et al.*,] W. Nutt, Y. Sagiv, and S. Shurin. Deciding equivalences among aggregated queries. DWQ Deliverable 7.1, 1997.

[Ross *et al.*, 1998] K. Ross, D. Srivastava, P. J. Stuckey, and S. Sudarshan. Foundations of aggregation constraints. *Theoretical Computer Science*, 1998. To appear.

[Schmidt-Schauß & Smolka, 1991] M. Schmidt-Schauß and G. Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48(1):1–26, 1991.

[Srivastava *et al.*, 1996] D. Srivastava, S. Dar, H. V. Jagadish, and A. Y. Levy. Answering queries with aggregation using views. In *Proceedings of the 22. International Conference on Very Large Data Bases (VLDB-96)*, Bombay, India, 1996.