RWTH
LTCS–Report

Aachen University of Technology
Research group for
Theoretical Computer Science

Description Logics with Aggregates and Concrete Domains

Franz Baader Ulrike Sattler

LTCS-Report 97-01

An abridged version has appeared in the Proceedings of the International Workshop on Description Logics, Gif sur Yvette, France, 1997.

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

Ahornstr. 55
52074 Aachen
Germany

Description Logics with Aggregates and Concrete Domains

Abstract

We show that extending description logics by simple aggregation functions as available in database systems may lead to undecidability of inference problems such as satisfiability and subsumption.

1 Motivation

Aggregation is a very useful mechanism available in many expressive representation formalisms such as database schema and query languages. Most systems provide a fixed set of aggregation functions like sum, min, max, average, count, which can be used over a given built-in domain, like the integers or the reals. In this paper, the generic Description Logic $\mathcal{ALC}(\mathcal{D})$ as introduced in [Baader & Hanschke,1991] is extended by aggregation. $\mathcal{ALC}(\mathcal{D})$ is an extension of the well-known description language $\mathcal{ALC}$ (see [Schmidt-Schauß & Smolka,1991; Hollunder *et al.*,1990; Donini *et al.*,1991]) by so-called *concrete domains*. In the basic language, $\mathcal{ALC}$, concepts can be built using propositional operators, (i.e., *and* ($\sqcap$), *or* ($\sqcup$), and *not* ($\neg$)), and value restrictions on those individuals associated to an individual via a certain role. These include *existential* restrictions like in ($\exists$has_child.Girl) as well as *universal* restrictions like ($\forall$has_child.Human). Additionally, in $\mathcal{ALC}(\mathcal{D})$, abstract individuals which are described using $\mathcal{ALC}$ can now be related to values in a *concrete domain* (e.g., the integers or strings) via *features*, i.e., functional roles. This allows us to describe managers that spend more money than they earn by

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

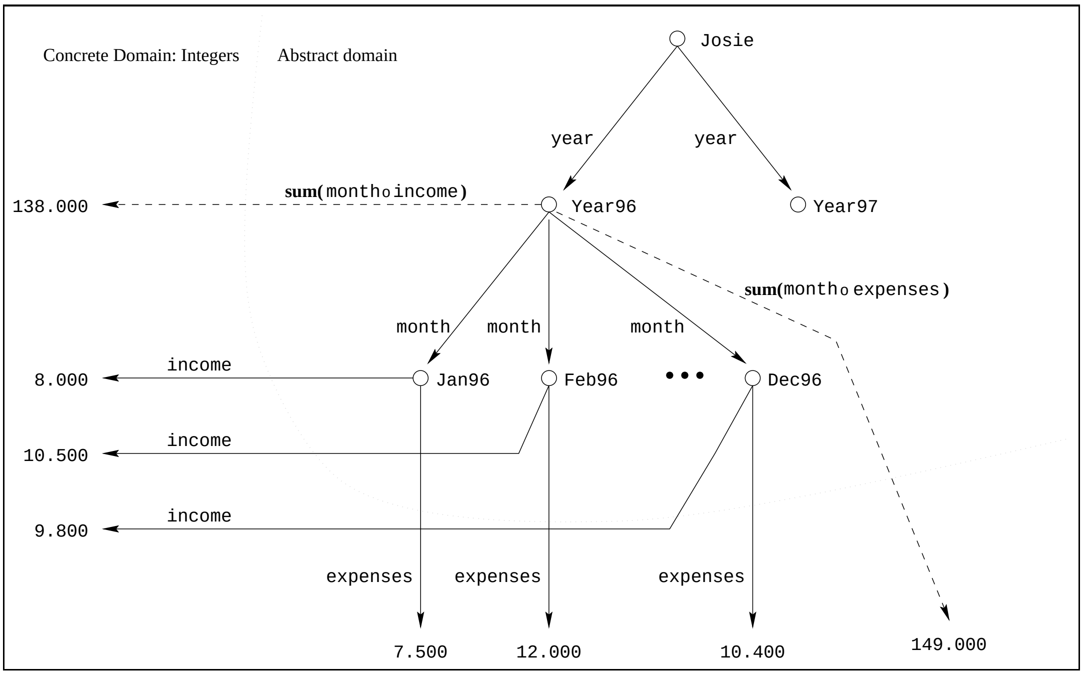

Figure 1: Example for aggregation

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

$$\begin{aligned}\mathtt{Human} \sqcap (\exists \mathtt{year}.\mathsf{less}(\,&\mathsf{sum}(\mathtt{month}\circ\mathtt{income}),\\ &\mathsf{sum}(\mathtt{month}\circ\mathtt{expenses}))),\end{aligned}$$

where the complex feature $\mathsf{sum}(\mathtt{month} \circ \mathtt{income})$ 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`.

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 turns out that, given a concrete domain together with aggregation functions satisfying some very weak conditions, this extension has an undecidable satisfiability problem. 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 a weak form of aggregation already leads to undecidability of satisfiability and subsumption.

For database research, these results are, for example, of interest 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]. They are not comparable with the undecidability results presented in [Mumick & Shmueli,1995] since our prerequisites are weaker and no recursion mechanisms are used. Neither are they contained in the undecidability results in [Ross *et al.*,1998]: the results presented there concern constraints involving multiplication and addition as well as rather complex aggregation functions like average or count—in contrast to the results presented here.

2 Preliminaries: 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 $\top$ 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.

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

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

where $(f_1 \circ \ldots \circ f_m)^\mathcal{I}(x) = f_m^\mathcal{I}(f_{m-1}^\mathcal{I}(\ldots(f_1^\mathcal{I}(x)\ldots)$. A concept C is called *satisfiable* iff there is some interpretation $\mathcal{I}$ such that $C^\mathcal{I} \neq \emptyset$. Such an

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

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 $\mathcal{I}$. Two concepts are said to be equivalent (written $C \equiv D$) if they mutually subsume each other. For an interpretation $\mathcal{I}$, an individual $x \in \Delta^\mathcal{I}$ is called an *instance* of a concept C iff $x \in C^\mathcal{I}$.

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 $x \in \top(f)^\mathcal{I}$, then $f^\mathcal{I}(x) \in \mathsf{dom}(\mathcal{D})$. To express that an individual has no f-successor at all, we will use the abbreviation $\mathsf{no}_f = \forall f.(A \sqcap \neg A)$. As $\mathcal{ALC}(\mathcal{D})$ allows for negation and conjunction of concepts, all boolean operators can be expressed, and we will use $C \Rightarrow D$ as a shorthand for $\neg C \sqcup D$. 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 $\mathsf{MS}(S)$.

A multiset M is said to be finite iff $\{s \mid M(s) \neq 0\}$ is a finite set.

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 $\mathcal{ALC}(\mathcal{D}+\Sigma)$-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 $\mathsf{dom}(\mathcal{D})$.

The set of *concrete features* is inductively defined as follows:

- Each feature name $f \in N_F$ is a concrete feature,
- chains of concrete features are concrete features,
- if $R \in N_R$ is a role, f is a concrete feature, $\Sigma \in \mathsf{agg}(\mathcal{D})$ is an aggregation function, then $\Sigma(R \circ f)$ is a concrete feature.

Finally, $\mathcal{ALC}(\mathcal{D}+\Sigma)$-concepts are obtained from $\mathcal{ALC}(\mathcal{D})$-concepts by allowing, additionally, the use of concrete features f_i in a predicate restrictions $P(f_1, \ldots, f_n)$ (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 $\mathcal{ALC}(\mathcal{D}+\Sigma)$-interpretation $\mathcal{I}$ is an $\mathcal{ALC}(\mathcal{D})$-interpretation which additionally satisfies

$$(\Sigma(R \circ f))^{\mathcal{I}} = \{(x, y) \in \Delta^{\mathcal{I}} \times \mathsf{dom}(\mathcal{D}) \mid \Sigma^{\mathcal{D}}(M_x^{R \circ f}) = y\}$$

where, for $x \in \Delta^{\mathcal{I}}$, a feature f, and a role R, $M_x^{R \circ f}$ denotes the multiset over $\mathsf{dom}(\mathcal{D})$ where the number of occurrences of $z \in M_x^{R \circ f}$ is determined by the number of $R^{\mathcal{I}}$-successors y of x with $f^{\mathcal{I}}(y) = z$, i.e. for $z \in \mathsf{dom}(\mathcal{D})$ we have

$$M_x^{R \circ f}(z) = \#\{y \in \Delta^{\mathcal{I}} \mid (x, y) \in R^{\mathcal{I}} \text{ and } f^{\mathcal{I}}(y) = z\}.$$

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

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

(b) Aggregation functions are *partial* functions, hence $(\Sigma(R \circ f))^{\mathcal{I}}(x)$ does not need to be defined. For example, the (standard) sum over an infinite set of numbers larger than 1 is undefined: If $\mathsf{dom}(\mathcal{D})$ is the set of reals, and if x has infinitely many R-successors in $\mathcal{I}$ which all have an f-successor in the reals that is larger than 1, then $(\mathsf{sum}(R \circ f))^{\mathcal{I}}(x)$ is undefined. To enforce that an individual has f_i-successors in $\mathsf{dom}(\mathcal{D})$, we can make use of predicate restrictions $P(f_1, \ldots, f_n)$. Recall that for $x \in \Delta^{\mathcal{I}}$ to be an instance of a concept $P(f_1, \ldots, f_n)$, it is necessary that for each concrete feature f_i the value $f_i^{\mathcal{I}}(x)$ is defined and in $\mathsf{dom}(\mathcal{D})$.

3.1 A first undecidability result

The following theorem states that admissibility of a concrete domain does no longer guarantee decidability of the interesting inference problems:

Theorem 7 For a concrete domain $\mathcal{D}$ where

- $\mathsf{dom}(\mathcal{D})$ includes the non-negative integers,
- $\mathsf{pred}(\mathcal{D})$ contains a (unary) predicate $P_{=1}$ that tests for equality with 1, and a (binary) equality $P_{=}$,
- $\mathsf{agg}(\mathcal{D})$ contains $\mathsf{min}, \mathsf{max}, \mathsf{sum}$,

satisfiability and subsumption of $\mathcal{ALC}(\mathcal{D} + \Sigma)$-concepts is undecidable.

Remarks: (a) The aggregation functions $\mathsf{min}, \mathsf{max}, \mathsf{sum}$ are supposed to be defined as usual, i.e., for multisets M over the reals (and thus also for multisets M over the non-negative integers) we have

$$\mathsf{sum}(M) = \begin{cases} \sum\limits_{y \in M} M(y) \cdot y & \text{if } M \text{ is finite} \\ \text{undefined} & \text{otherwise} \end{cases}$$

$$\mathsf{min}(M) = \begin{cases} m & \text{if there exists } m \in M \text{ such} \\ & \text{that } n \geq m \text{ for all } n \in M \\ \text{undefined} & \text{if such an } m \text{ does not exist} \end{cases}$$

$$\mathsf{max}(M) = \begin{cases} m & \text{if there exists } m \in M \text{ such} \\ & \text{that } n \leq m \text{ for all } n \in M \\ \text{undefined} & \text{if such an } m \text{ does not exist} \end{cases}$$

(b) At first sight, this undecidability result seems very restricted. Note, however, that it does not require that $\mathsf{dom}(\mathcal{D})$ is the set of non-negative integers, but that it just requires that $\mathsf{dom}(\mathcal{D})$ *contains* the non-negative integers. This makes the undecidability result not only more general, but also stronger: For example, computations over the reals are, in general, easier than computations over the non-negative integers, i.e., the first order theory of $+, \cdot, \leq$ is undecidable over the non-negative integers, whereas it is decidable over the reals.

Furthermore, the aggregation functions $\mathsf{min}, \mathsf{max}, \mathsf{sum}$ are among those normally considered as built-in functions for databases (see, for example, [Gupta *et al.*,1995; Mumick & Shmueli,1995; Levy & Mumick,1996; Srivastava *et al.*,1996]). Finally, to test whether a certain value equals 1 or whether two values are equal is possible in all database systems with built-in predicates.

(c) We do not suppose that $\mathcal{D}$ is admissible—although this precondition would not make the undecidability result less expressive. Nevertheless, in the sequel we will make use of the concept $\top(f)$. This is in accordance with the preconditions of Theorem 7 because $\top(f)$ (if not available in $\mathcal{D}$) can be introduced as abbreviation, e.g., for $P_=(f, f)$.

Proof of Theorem 7: The proof is by reduction of Hilbert's 10th problem [Davis,1973] to satisfiability of concepts, i.e., for polynomials $P, Q \in \mathbb{N}[x_1, \ldots, x_m]$, one can construct an $\mathcal{ALC}(\mathcal{D} + \Sigma)$-concept $C_{P,Q}$ that is satisfiable iff the polynomial equation

$$P(x_1, \ldots, x_m) = Q(x_1, \ldots, x_m) \tag{1}$$

has a solution in $\mathbb{N}^m$. In the sequel, we write $\mathbf{x}$ as shorthand for $(x_1, \ldots, x_m)$ and $\mathbf{x}^{\mathbf{i}_j}$ as shorthand for the monomial $x_1^{i_{j1}} \cdots x_m^{i_{jm}}$.

The idea of the reduction is to represent the (sub)term structure of the polynomial P (resp. Q) as a tree which is related to an instance of $C_{P,Q}$ via the feature P (resp. Q); see Figure 2. The polynomial P is supposed to be of the form

$$P(\mathbf{x}) = a_0 + a_1\mathbf{x}^{\mathbf{i}_1} + \ldots + a_j\mathbf{x}^{\mathbf{i}_j} + \ldots a_n\mathbf{x}^{\mathbf{i}_n},$$

where all monomials $\mathbf{x}^{\mathbf{i}_j}$ are supposed to be different.

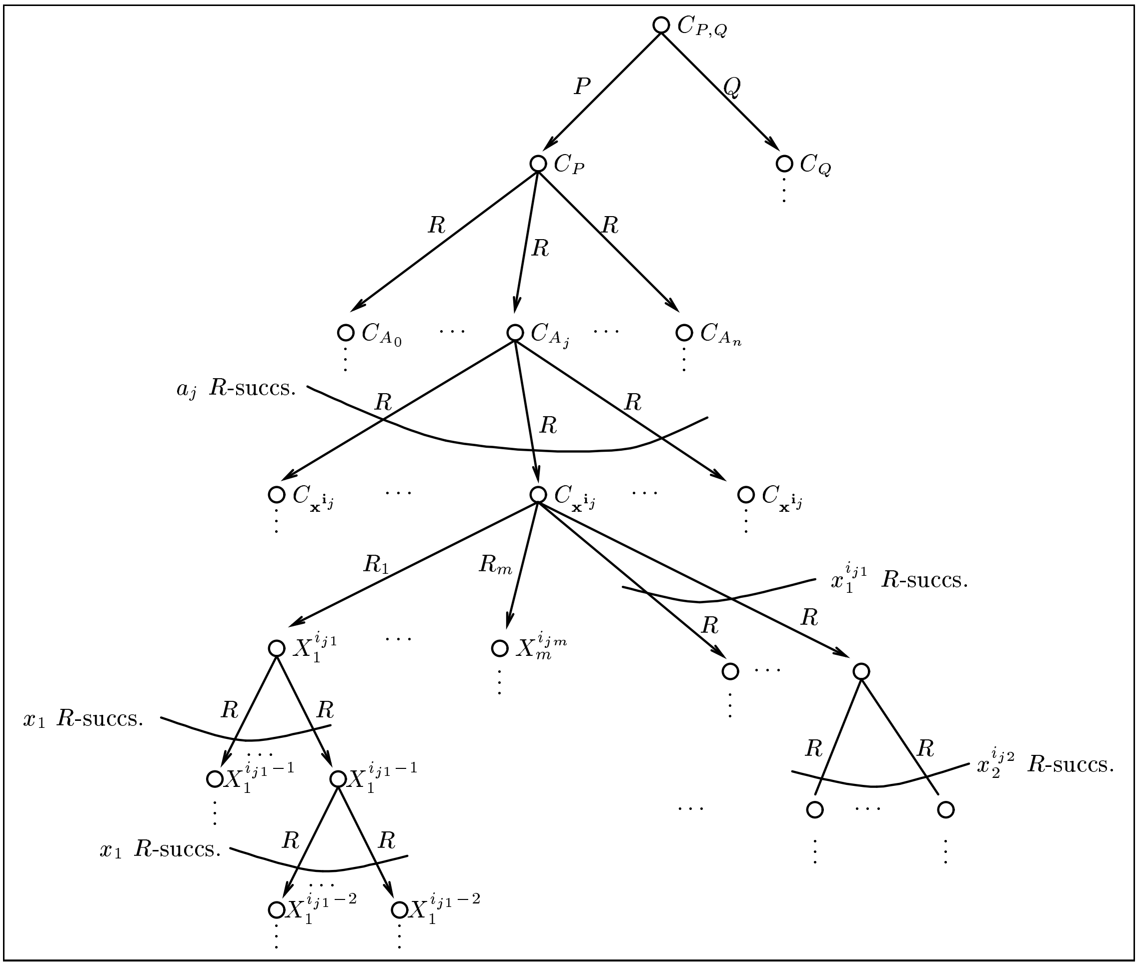

Figure 2: The intuitive structure of $C_{P,Q}$

When building the reduction concept $C_{P,Q}$, one encounters three main problems: (a) We only know that $\mathsf{dom}(\mathcal{D})$ *contains* $\mathbb{N}$, but the solution of Equation 1 has to be in $\mathbb{N}^m$, and $\mathcal{D}$ need not provide for a predicate that tests for being a non-negative integer. (b) It has to be assured that (the representation of) each variable x_i is associated with the same non-negative integer wherever it occurs in a model of $C_{P,Q}$. (c) The reduction asks for the representation of calculations such as addition, multiplication, and exponentiation.

These problems can be overcome as follows:

(a) is solved by making use of the concept E_g^R,

$$E_g^R := (\forall R.(P_{=1}(f))) \sqcap P_{=}(\mathsf{sum}(R \circ f), g),$$

whose instances have as g-successors the number of their R-successors. Hence their g-successor is in $\mathbb{N}$ or undefined (if there are infinitely many R-successors).

(b) This problem is solved by introducing features $\mathtt{x}_i$ for each variable x_i and by making strong use of the concepts $E_{\mathtt{x}_i}^R$ as defined above and Inv:

$$\mathsf{Inv} := \bigsqcap_{1 \leq i \leq m} (\forall R.\top(\mathtt{x}_i) \sqcap P_{=}(\mathsf{min}(R \circ \mathtt{x}_i), \mathsf{max}(R \circ \mathtt{x}_i)) \sqcap P_{=}(\mathtt{x}_i, \mathsf{max}(R \circ \mathtt{x}_i))).$$

Using $E_{\mathtt{x}_i}^R$, we make sure that $\mathtt{x}_i$-successors are non-negative integers. The concept Inv is defined in such a way that R-successors of an instance a of Inv have the same $\mathtt{x}_i$-successor, which coincides with the $\mathtt{x}_i$-successor of a.

Using this concept at all levels of nested concepts, we can guarantee that all "relevant" individuals in a model of $C_{P,Q}$ have the same $\mathtt{x}_i$-successor for each variable x_i.

(c) Addition can be realized by the aggregation function sum, and multiplication (and hence exponentiation) can be reduced to addition; for details see the explanation of the reduction concepts below.

Let $\mathcal{D}$ be as described in Theorem 7. Then we can define the following abbreviations:

$$E_1^R := (\forall R.(P_{=1}(f))) \sqcap P_{=1}(\mathsf{sum}(R \circ f)) \quad \text{(exactly 1 } R\text{-successor)}$$

$$E_g^R := (\forall R.(P_{=1}(f))) \sqcap P_{=}(\mathsf{sum}(R \circ f), g) \quad \text{(exactly } g^{\mathcal{I}}(x) \text{ } R\text{-successors)}$$

$$E_n^R := \forall R.\left(\bigsqcup_{1 \leq i \leq n} (P_{=1}(f_i) \sqcap \bigsqcap_{j \neq i} \mathsf{no}_{f_j}\right) \sqcap \bigsqcap_{1 \leq i \leq n} P_{=1}(\mathsf{sum}(R \circ f_i)) \quad \text{(exactly } n \text{ } R\text{-successors)}$$

where no_{f_j} is the abbreviation for $\forall f_j.(A \sqcap \neg A)$ mentioned in Section 2.

It is easy to see that each instance of E_1^R has exactly 1 R-successor, and the concept E_g^R has already been explained above. Now, for an instance a of E_n^R, every R-successor has exactly one f_i-successor for some $i, 1 \leq i \leq n$, and this f_i-successor has value 1 (first line). The constraint on the concrete feature $\mathsf{sum}(R \circ f_i)$ (second line) makes sure that there is exactly one R-successor with an f_i-successor for each i, which implies that a has exactly n R-successors.

More precisely, the reduction concept is built as follows and given in the Figures 3 and 4.

1. First, we define $C_{P,Q}$ such that, for each interpretation $\mathcal{I}$, each instance $x \in C_{P,Q}^{\mathcal{I}}$ has exactly one P-successor p in $C_P^{\mathcal{I}}$ and exactly one Q-successor q in $C_Q^{\mathcal{I}}$. The individual p represents the polynomial P, and q represents Q; see Concept 2. Concept 3 is similar to Inv and makes sure that for each j, the $\mathtt{x}_j$-successor of p is in $\mathsf{dom}(\mathcal{D})$ and the same as the $\mathtt{x}_j$-successor of q. Finally, Concept 4 makes sure that the value of the polynomial P when evaluated with the $\mathtt{x}_j$-successors (which are already ensured to be the same for p and for q) is the same as of Q.

2. An instance p of C_P has

   - for each monomial $A_j = a_j \mathbf{x}^{\mathbf{i}_j}$ of P one R-successor which is an instance of C_{A_j} and represents the monomial A_j; see Concept 5.
   - an s-successor which is the sum of the s-successors of its R-successors; see Concept 6.

   Given that the s-successor of each R-successor of p is the value of the monomial A_j, the s-successor of p is the corresponding value of P, namely the sum over all monomials. Again, the concept Inv makes sure that each $\mathtt{x}_i$-successor of p coincides with the $\mathtt{x}_i$-successors of its R-successors.

3. For the monomials A_j, we use $n+1$ concepts C_{A_j}. The purpose of the last conjunct of Concept 7 is to achieve disjointness of these concepts C_{A_j}. An instance a of C_{A_j} has a_j R-successors, each of them representing $\mathbf{x}^{\mathbf{i}_j}$; see Concept 9. The last conjunct makes sure that the s-successor (representing the value of A_j) is computed correctly: Since

$$C_{P,Q} := E_1^P \sqcap E_1^Q \sqcap \forall P.C_P \sqcap \forall Q.C_Q \sqcap \quad (2)$$

$$\bigsqcap_{1\le j\le m} \Big(P_=(\mathsf{sum}(P\circ \mathtt{x}_j), \mathsf{sum}(Q\circ \mathtt{x}_j)) \Big) \sqcap \quad (3)$$

$$P_=(\mathsf{sum}(P\circ s), \mathsf{sum}(Q\circ s)) \quad (4)$$

$$C_P := E_{n+1}^R \sqcap \bigsqcap_{0\le i\le n} (\exists R.C_{A_i}) \sqcap \quad (5)$$

$$\mathsf{Inv} \sqcap P_=(s, \mathsf{sum}(R\circ s)) \quad (6)$$

$$C_{A_j} := E_{a_j}^R \sqcap \forall R.C_{\mathbf{x}^{\mathbf{i}_j}} \sqcap E_j^H \sqcap \quad (7)$$

$$\mathsf{Inv} \sqcap P_=(s, \mathsf{sum}(R\circ s)) \quad (8)$$

$$C_{\mathbf{x}^{\mathbf{i}_j}} := \mathsf{Exp}_{\mathbf{x}^{\mathbf{i}_j}} \sqcap \mathsf{Mult}_1^m \quad (9)$$

Figure 3: The reduction concept $C_{P,Q}$ and some of its subconcepts.

a has a_j R-successors, each of them representing $\mathbf{x}^{\mathbf{i}_j}$, the s-successor of a is the sum over the s-successors of its R-successors.

4. $C_{\mathbf{x}^{\mathbf{i}_j}}$ is more complicated. An instance c of it has two different kinds of role successors:

   - For each of the m factors $x_k^{i_{jk}}$ in $\mathbf{x}^{\mathbf{i}_j}$, c has one R_k-successor in $X_k^{i_{jk}}$, whose s_k-successor stands for its value $x_k^{i_{jk}}$. The concept $\mathsf{Exp}_{\mathbf{x}^{\mathbf{i}_j}}$ implies this fact. In $\mathsf{Exp}_{\mathbf{x}^{\mathbf{i}_j}}$, we use the second conjunct instead of Inv to propagate the value of x_k down to the according subtree. The last conjunct of $\mathsf{Exp}_{\mathbf{x}^{\mathbf{i}_j}}$ makes sure that the respective values s_k are propagated upwards to c.
   - Then, in order to multiply the m factors $x_k^{i_{jk}}$, we make use of the concept Mult_1^m explained below.

$$\mathsf{Exp}_{\mathbf{x}^{\mathbf{i}_j}} := \prod_{1\le k\le m}\Big(E_1^{R_k} \sqcap P_=(\mathtt{x}_k, \mathsf{sum}(R_k \circ \mathtt{x}_k)) \sqcap \tag{10}$$

$$\forall R_k.X_k^{i_{jk}} \sqcap P_=(s_k, \mathsf{sum}(R_k \circ s_k))\Big) \tag{11}$$

$$\mathsf{Mult}_m^m := P_=(s, s_m) \tag{12}$$

$$\mathsf{Mult}_k^m := E_{s_k}^R \sqcap P_=(s, \mathsf{sum}(R \circ s)) \sqcap \forall R.\mathsf{Mult}_{k+1}^m \sqcap \tag{13}$$

$$\prod_{\ell=k+1}^{m}\Big(P_=(\mathsf{min}(R \circ s_\ell), \mathsf{max}(R \circ s_\ell)) \sqcap P_=(\mathsf{min}(R \circ s_\ell), s_\ell)\Big) \tag{14}$$

$$X_k^0 := P_{=1}(s) \sqcap E_{\mathtt{x}_k}^R \tag{15}$$

$$X_k^1 := E_{\mathtt{x}_k}^R \sqcap P_=(s, \mathtt{x}_k) \tag{16}$$

$$X_k^\ell := E_{\mathtt{x}_k}^R \sqcap \forall R.X_k^{\ell-1} \sqcap P_=(s, \mathsf{sum}(R \circ s)) \sqcap \tag{17}$$
$$P_=(\mathsf{min}(R \circ \mathtt{x}_k), \mathsf{max}(R \circ \mathtt{x}_k)) \sqcap P_=(\mathtt{x}_k, \mathsf{max}(R \circ \mathtt{x}_k)),\ \ \ell \geq 2$$

Figure 4: Subconcepts of $C_{P,Q}$ used for the representation of calculations.

Again, the s-successor of c denotes the value of this calculation, namely $\mathbf{x}^{\mathbf{i}_j}$.

5. For X_k^i, we have to distinguish two cases : If $i = 0$, then the value associated to this factor is 1; see the concept X_k^0. Otherwise, an instance y of X_k^i is the root of an x_k-ary R-tree of depth i where the s-successor of each node is the sum of the s-successors of its R-successors. Finally, the s-successor of a node one level above the leaves (which represents x_k^1) equals its $\mathtt{x}_k$-successor—which is the same for all nodes in the whole tree. Since $\mathsf{dom}(\mathcal{D})$ is only required to contain the non-negative integers, we have to ensure that all $\mathtt{x}_k$-successors are non-negative integers. This is realized by making use of the concept $E_{\mathtt{x}_k}^R$. Thus, we use the possibilities to construct trees and to sum up in order to compute ex-

ponentiation.

6. Finally, the situation in which we start multiplication looks as follows: An instance u of Mult_1^m is at the root of the multiplication tree, u is also an instance of $C_{\mathbf{x}^{\mathbf{i}_j}}$, and we want to multiply all m s_k-successors of u. To this purpose, we attach an R-tree of depth $m-1$ to u. This tree is, at level k, of outdegree s_k. At level $m-1$, we make sure that the s_m-successors coincide with the s-successor. Again, we sum up the values from the bottom to the top by using the concept $P_=(s, \mathsf{sum}(R \circ s))$, and we make sure that all nodes have the same s_i successor by a concept similar to Inv; see Concept 14.

It remains to be shown that $C_{P,Q}$ is satisfiable iff $P(\mathbf{x}) = Q(\mathbf{x})$ admits a solution in the non-negative integers.

"⇐" The construction of a model M for $C_{P,Q}$ from P, Q, and a solution $n_1, \ldots, n_m \in \mathbb{N}^m$ for $\mathbf{x}$ is not difficult. M can be constructed along the explanations given for $C_{P,Q}$ in the following way: We start at the bottom of the tree M by introducing instances x_k^1 of

- X_k^1 that have n_k R-successors, each of them having 1 as f-successor (due to the use of $E_{\mathtt{x}_k}^R$), n_k as $\mathtt{x}_k$ successor, and n_k as s-successor, and instances x_k^0 of
- X_k^0 that have n_k R-successors, each of them having 1 as f-successor, n_k as $\mathtt{x}_k$ successor, and 1 as s-successor.

Then, for each monomial $\mathbf{x}^{\mathbf{i}_j}$, the corresponding subtrees representing $n_k^{i_{jk}}$ are built: Starting with (copies of) x_k^1 and x_k^0, we build trees of depth i_{jk} and degree n_k. Next, instances c of $C_{\mathbf{x}^{\mathbf{i}_j}}$ are introduced, where each c has as R_k-successor the subtree representing the factor $n_k^{i_{jk}}$ in $n_1^{i_{j1}} \cdots n_m^{i_{jm}}$. Now, we have to append another subtree to each c, namely the one representing the multiplication of the values $n_k^{i_{jk}}$. This tree is of depth $m-1$ and degree $n_k^{i_{jk}}$ at level k. The remaining construction is straightforward: We first take a_j disjoint copies of the c's standing for $C_{\mathbf{x}^{\mathbf{i}_j}}$ (including the corresponding subtree) as R-successors of an instance a of C_{A_j}, then we append these a's as

R-successors of an instance of p of C_P. We suppose that the same construction has been carried out for Q, which lead to an instance q of C_Q. Finally, p and q are P (resp. Q) -successors of an instance c of $C_{P,Q}$.

All over the tree constructed in this way, the s-successor of an individual equals the sum over the s-successors of its R-successors, and all individuals have the same $\mathtt{x}_k$-successor. The fact that a solution $n_1, \ldots, n_m \in \mathbb{N}^m$ for $\mathbf{x}$ has to be used for this construction is reflected in the fact that, due to the definition of $C_{P,Q}$, p's s-successor has to coincide with q's s-successor.

"$\Rightarrow$" Given a model M for $C_{P,Q}$ with $c \in C^{\mathcal{I}}_{P,Q}$, due to the presence of Inv and similar concepts in $C_{P,Q}$, all $\mathtt{x}_i$-successors of all "relevant" role successors of c coincide—where "relevant" role successors are those whose existence is explicitly required by $C_{P,Q}$. Again, following the description of $C_{P,Q}$, it is easy to see that $(\mathtt{x}_1^{\mathcal{I}}(c), \ldots, \mathtt{x}_m^{\mathcal{I}}(c))$ is a solution for $P(\mathbf{x}) = Q(\mathbf{x})$. Due to the use of the concepts $E^R_{\mathtt{x}_i}$, this solution is in $\mathbb{N}^m$.

Hence satisfiability and thus subsumption of $\mathcal{ALC}(\mathcal{D})$-concepts is undecidable for concrete domains $\mathcal{D}$ as described in Theorem 7. ■

We want to emphasize that $C_{P,Q}$ does not make any use of the possibility to apply aggregation functions to feature chains, i.e., wherever a subconcept of $C_{P,Q}$ contains $\Sigma(R \circ f)$ for some aggregation function Σ, f is a feature name (and not a complex feature chain or concrete feature).

3.2 Tightening the result

A closer investigation of the concept $C_{P,Q}$ reveals that (a) negation occurs only in the concept no_f, (b) the only place where existential restriction occurs is in the concepts C_P and C_Q, and (c) the only place where disjunction $\sqcup$ occurs is in the concepts E^R_n describing individuals having exactly n R-successors.

It can be shown that the concepts no_f, E^R_n and C_P can be rewritten into concepts without negation, disjunction and existential restriction, by extending only slightly the set of concrete predicates. Hence, the reduction concept $C_{P,Q}$ can be written using only conjunction $\sqcap$ and universal value restriction

$\forall R.C$. As introduced in [Baader,1996], let $\mathcal{FL}_0$ denote the set of those concepts that are built using conjunction and universal value restriction only, and let $\mathcal{FL}_0(\mathcal{D}+\Sigma)$ denote the extension of this language by concrete domains with aggregation. Then the following undecidability result is an immediate consequence of the possibility to rewrite the reduction concept $C_{P,Q}$ without using negation, disjunction, and existential restriction.

Theorem 8 For a concrete domain $\mathcal{D}$ where

- $\mathsf{dom}(\mathcal{D})$ includes the non-negative integers $\mathbb{N}$,
- $\mathsf{pred}(\mathcal{D})$ contains, for all non-negative integers n, (unary) predicates $P_{=n}$ that test for equality with n, the (binary) equality predicate $P_=$, and the (binary) inequality predicate $P_{\neq}$,
- $\mathsf{agg}(\mathcal{D})$ contains $\mathsf{min}, \mathsf{max}, \mathsf{sum}$,

satisfiability and subsumption of $\mathcal{FL}_0(\mathcal{D}+\Sigma)$-concepts is undecidable.

Remarks: (a) Admissible concrete domains as defined in [Baader & Hanschke,1991] are closed under negation, hence the presence of a predicate $P_=$ in $\mathsf{pred}(\mathcal{D})$ implies the presence of its negation $P_{\neq}$. Thus, for admissible domains, only the unary predicates $P_{=n}$ are required in addition to the preconditions of Theorem 7.

(b) We recall that according to the semantics of $\mathcal{FL}_0(\mathcal{D}+\Sigma)$, an individual x can only be an instance of the concept $P_{\neq}(f,g)$ if x has an f- as well as a g-successor in the concrete domain $\mathsf{dom}(\mathcal{D})$.

Proof: It remains to define $\mathcal{FL}_0(\mathcal{D}+\Sigma)$-concepts no'_f, ${E'}_n^R$ and C'_P which can play the rôle of no_f, E_n^R and C_P in the reduction concept $C_{P,Q}$ of the proof of Theorem 7.

$\mathsf{no'_f}$: This concept is used to make sure that an individual has no f-successor. It can be clearly replaced by

$$\mathsf{no}'_f := \forall f.P_{\neq}(g,g),$$

where $P_{\neq}(g,g)$ plays the rôle of the empty concept $A \sqcap \neg A$ used in the definition of no_f.

$\mathbf{E'^{\,R}_n}$: Given a concrete domain $\mathcal{D}$ that provides, for all non-negative integers n, a unary predicate $P_{=n}$ that tests for equality with n, we can define a concept ${E'}_n^R$ whose instances have exactly n R-successors:

$$ {E'}_n^R := \forall R.P_{=1}(f) \sqcap P_{=n}(\mathsf{sum}(R \circ f)). $$

Obviously, replacing E_n^R by ${E'}_n^R$ in $C_{P,Q}$ preserves its property of serving as a reduction concept for Hilbert's 10th problem. Avoiding existential restriction in C_P is more complicated.

$\mathbf{C'_P}$: In C_P, existential restriction is used to make sure that for each monomial A_j there is one R-successor representing this monomial (the uniqueness of this R-successor stems from the fact that there are exactly $n+1$ R-successors and that the C_{A_j} are mutually disjoint). This can also be expressed by introducing for each j exactly one R_j-successor (using $E_1^{R_j}$), and then using universal value restrictions to make sure that the R_j-successor is an instance of C_{A_j}. Additionally, the $\mathtt{x}_j$-successors have to be propagated to the R_j-successors. All this is ensured by the first line of C'_P.

$$ \begin{aligned} C'_P := \; & \bigsqcap_{0 \le j \le n} \Big(E_1^{R_j} \sqcap \forall R_j.C_{A_j} \sqcap \bigsqcap_{0 \le \ell \le m} P_{=}(\mathtt{x}_\ell, \mathsf{sum}(R_j \circ \mathtt{x}_\ell)) \sqcap \\ & \qquad P_{=}(s_j, \mathsf{sum}(R_j \circ s)) \Big) \sqcap \\ & \mathsf{Add}_{s_0,\ldots,s_n} \end{aligned} $$

It remains to enforce that the sum of all s-successors of all R_j-successors of an instance p of C'_P coincides with p's s-successor. For this purpose, we make sure that p has an s_j-successor which coincides with the s-successor of its R_j-successor. Then the concept $\mathsf{Add}_{s_0,\ldots,s_n}$ is used to sum up p's s_j-successors. It is defined inductively as follows:

$$ \begin{aligned} \mathsf{add}^t_{s_0,s_1} := \; & \forall R_{01}.(E_{g_0}^{G_0} \sqcap E_{g_1}^{G_1} \sqcap P_{\neq}(g_0, g_1)) \sqcap \\ & P_{=1}(\mathsf{max}(R_{01} \circ g_0)) \sqcap P_{=0}(\mathsf{min}(R_{01} \circ g_0)) \sqcap \\ & P_{=1}(\mathsf{max}(R_{01} \circ g_1)) \sqcap P_{=0}(\mathsf{min}(R_{01} \circ g_1)) \sqcap \\ & P_{=}(s_0, \mathsf{sum}(R_{01} \circ g_0)) \sqcap P_{=}(s_1, \mathsf{sum}(R_{01} \circ g_1)) \sqcap \\ & E_t^{R_{01}} \end{aligned} $$

The idea underlying this addition is the following. First, the addition of $n+1$ numbers is reduced to the addition of two numbers: Therefore, the s_0- and the s_1-successor of p are summed up and the result is stored as s_{01}-successor of p. Next, the s_{01}- and the s_2-successor are summed up and the result is stored as s_{012}-successor of p, and so forth, until only two arguments are left. The sum of these last numbers is the result of the whole addition, and therefore stored as s-successor of p.

The addition of two numbers (given as s_0- and s_1-successors) and the storage of the result as t-successor is realized by the concept $\mathsf{add}^t_{s_0,s_1}$ given above: Let p be an instance of $\mathsf{add}^s_{s_0,s_1}$, let $s0$ be p's s_0-successor, and let $s1$ be p's s_1-successor. From the construction of $C_{P,Q}$ it follows that $s0, s1$ are non-negative integers. We make sure that the number of R_{01}-successors of p equals $s0+s1$. Additionally, p's t-successor equals the number of its R_{01}-successors, which is $s0+s1$.

To enforce that the number of R_{01}-successors of p equals $s0+s1$, we need all but the last line of concept $\mathsf{add}^s_{s_0,s_1}$. The idea is to partition the R_{01}-successors into those contributing to $s0$ and those contributing to $s1$. R_{01}-successors contributing to $s0$ have 1 as g_0-successor successor and 0 as g_1-successor, whereas R_{01}-successors contributing to $s1$ have 0 as g_0-successor and 1 as g_1-successor. The first line of $\mathsf{add}^s_{s_0,s_1}$ ensures that all g_i-successors of all R_{01}-successors of p are non-negative integers (by using auxiliary roles G_0, G_1) and that the g_0-successor always differs from the g_1-successor. The next two lines make sure that g_i-successors of R_{01}-successors are between 0 and 1. Together with the fact that they are non-negative integers and different, we have that each R_{01}-successor has either 1 as g_0-successor and 0 as g_1-successor or vice versa. The fourth line states that the number of R_{01}-successors representing $s0$ (i.e., the ones having 1 as g_0-successor) is $s0$, and that the number of R_{01}-successors representing $s1$ is $s1$. Finally, the last line enforces that the number of p's R_{01}-successors coincides with its s-successor.

Again, replacing C_P by C'_P (respectively C_Q by C'_Q) in $C_{P,Q}$ preserves its property of serving as a reduction concept for Hilbert's 10th problem, which is—in contrast to the initial one—an $\mathcal{FL}_0(\mathcal{D}+\Sigma)$-concept.

Undecidability of subsumption follows from undecidability of satisfiability because a concept C is satisfiable iff it is not subsumed by an unsatisfiable concept, and because the $\mathcal{FL}_0(\mathcal{D}+\Sigma)$-concept $P_{\neq}(f,f)$ is such an unsatisfiable concept. ■

4 Conclusion

Reasoning with constraints involving aggregation functions is a crucial task for many advanced information systems like decision support and on-line-analytical processing systems, data warehouses, and (statistical) databases [Ross *et al.*,1998; Gupta *et al.*,1995; Mumick & Shmueli,1995; De Giacomo & Naggar,1996; Levy & Mumick,1996; Srivastava *et al.*,1996]. The more the amount of data that are processed by these systems grows, the more important become aggregation functions for summarizing, consolidating and analyzing these large amounts of data. Hence, traditional techniques for query rewriting, query optimization, view maintenance, etc. must be extended such that they are able to cope with aggregation functions.

The two undecidability results presented in this paper indicate that this task will be difficult. The aggregation functions $\mathsf{min}, \mathsf{max}, \mathsf{sum}$ that suffice to obtain undecidability are the most "well-behaved" ones: aggregation functions like count or $\mathsf{average}$ are much more difficult to handle. For example, $\mathsf{min}, \mathsf{max}, \mathsf{sum}$ are monotonic, i.e., if $S \subseteq S'$, then

$$\begin{aligned} \mathsf{min}(S) &\geq \mathsf{min}(S'), \\ \mathsf{max}(S) &\leq \mathsf{max}(S'), \\ \mathsf{sum}(S) &\leq \mathsf{sum}(S'), \end{aligned}$$

whereas these relations cannot be established for count or $\mathsf{average}$. Furthermore, they are "compositional" in the sense that the aggregation $f \in \{\mathsf{min}, \mathsf{max}, \mathsf{sum}\}$ of two disjoint multisets S, S' can be computed using f, $f(S)$, $f(S')$ only—which does not hold, for example, for $\mathsf{average}$. Hence, our undecidability result cannot be said to be caused by using a too powerful set of aggregation functions.

Arguing from another perspective, $\mathcal{ALC}(\mathcal{D} + \Sigma)$ is a rather expressive Description Logic and it might not be very surprising that adding aggregation to $\mathcal{ALC}(\mathcal{D})$ leads to undecidability. In contrast, $\mathcal{FL}_0$ is, to our knowledge, the weakest Description Logic ever considered. It is of such a low expressive power that subsumption between two $\mathcal{FL}_0$-concepts can be reduced to answering conjunctive queries: given two $\mathcal{FL}_0$-concepts C_1 and C_2, C_1 subsumes C_2 if and only if an individual x of an extensional database $\mathsf{edb}_{C_1(x)}$ constructed from C_1 is in the answer set of a conjunctive query q_{C_2} constructed from C_2. This reduction is, for several reasons, not possible for

$\mathcal{FL}_0(\mathcal{D}+\Sigma)$-concepts. However, it leads to the speculation that (intensional) reasoning for conjunctive queries with (simple) aggregation functions and built-in predicates is of high computational complexity.

References

[Baader, 1996] F. Baader. Using automata theory for characterizing the semantics of terminological cycles. *Annals of Mathematics and Artificial Intelligence*, 18(2–4):175–219, 1996.

[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.

[Davis, 1973] M. Davis. Hilbert's tenth problem is unsolvable. *American Mathematical Monthly*, 80:233–269, 1973.

[De Giacomo & Naggar, 1996] G. De Giacomo and P. Naggar. Conceptual data model with structured objects for statistical databases. In *Proceedings of the Eighth International Conference on Statistical Database Management Systems (SSDBM'96)*, pages 168–175. IEEE Computer Society Press, 1996.

[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 (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.

[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.

[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.