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 for a fixed set of aggregation functions like \( \text{sum, min, max, average, count} \), which can be used over a given built-in domain like the integers or the reals. In this paper, the generic Description Logic \( \mathcal{ALC}(D) \), as introduced in [Baader & Hanschke1991], is extended by aggregation. \( \mathcal{ALC}(D) \) is an extension of the well-known description language \( \mathcal{ALC} \) (see [Schmidt-Schauss & Smolka1991; Hollunder et al.1990; Donini et al.1991]) by a so-called \emph{concrete domain}. In the basic language \( \mathcal{ALC} \), concepts can be built using propositional operators, (i.e., \( \land \), \( \lor \), or \( \neg \)), and value restrictions on those individuals associated to an individual via a certain role. These are \emph{existential} restrictions like in \( \exists \text{has\_child}\text{Girl} \) as well as \emph{universal} restrictions like \( \forall \text{has\_child}\text{Human} \). Additionally, in \( \mathcal{ALC}(D) \), abstract individuals, which are described using \( \mathcal{ALC} \), can be related to values in a \emph{concrete domain} (e.g., the integers or strings) via \emph{features}, i.e., functional roles. This allows us to describe managers that spend more money than they earn by \( \text{Manager} \cap (\exists \text{income, expenses}) \). In our extension of \( \mathcal{ALC}(D) \), 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

\[
\text{Human} \cap (\exists \text{year, less}(\text{sum(month\_income)}, \text{sum(month\_expenses)})).
\]

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

In this paper, we present a generic extension of \( \mathcal{ALC}(D) \) 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; Munick & Shmueli1995; Levy & Mumick1996; Srivastava et al.1996]. They are not comparable with the undecidability results presented in [Munick & Shmueli1995] 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 \( \text{average} \) or \( \text{count} \)—in contrast to the results presented here.

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

Before we can introduce \( \mathcal{ALC}(D) \), as defined in [Baader & Hanschke1991], we must specify the notion of a concrete domain.

Definition 1

A \emph{concrete domain} \( D = (\text{dom}(D), \text{pred}(D)) \) consists of

\( \text{supported by the DFG under Grant No. Sp 230}\ 6-6 \) and by the EU Working Group DWQ,
a set \( \text{dom}(D) \) (the domain) and a set of predicate symbols \( \text{pred}(D) \). Each predicate symbol \( P \in \text{pred}(D) \) is associated with an arity \( n \) and an \( n \)-ary relation \( P^D \subseteq \text{dom}(D)^n \).

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 \( D \) is called admissible if (a) \( \text{pred}(D) \) is closed under negation and contains a unary predicate name \( \top \) for \( \text{dom}(D) \), and (b) satisfiability in \( D \) of finite conjunctions over \( \text{pred}(D) \) is decidable. The syntax of \( \mathcal{ALC}(D) \)-concepts is now defined as follows:

**Definition 2** Let \( N_c, N_R, \) and \( N_F \) be disjoint sets of concept, role, and feature names. The set of \( \mathcal{ALC}(D) \)-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, \( P \in \text{pred}(D) \) is a 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 \( \text{dom}(D) \), called the domain of \( \mathcal{I} \), and a function \( \cdot ^\mathcal{I} \) that 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 \( f^\mathcal{I} : \Delta^\mathcal{I} \rightarrow \Delta^\mathcal{I} \cup \text{dom}(D) \). Furthermore, \( \mathcal{I} \) has to satisfy the following properties

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

where \((f_1 \circ \cdots \circ f_m)^\mathcal{I}(x) := f_m^\mathcal{I}(\cdots (f_2^\mathcal{I}(f_1^\mathcal{I}(x)) \cdots)\). A concept \( C \) is called satisfiable if there is some interpretation \( \mathcal{I} \) such that \( C^\mathcal{I} \neq \emptyset \). A concept \( D \) is said to subsume another concept \( C \) (written \( C \subseteq D \)) iff all interpretations \( \mathcal{I} \) satisfy \( C^\mathcal{I} \subseteq D^\mathcal{I} \). For an interpretation \( \mathcal{I} \), an individual \( x \in \Delta^\mathcal{I} \) is called an instance of a concept \( C \) if \( x \in C^\mathcal{I} \).

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

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

In order to define aggregation appropriately, first, we will introduce the notion of multisets: in contrast to

![Figure 1: Example for aggregation](image-url)
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** 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 \( M \). We write \( s \in M \) as shorthand for \( M(s) \geq 1 \). A multiset \( M \) is said to be finite if \( \{ s \mid M(s) \neq 0 \} \) is a finite set.

As the aggregation functions depend strongly on the specific concrete domains, the notion of a concrete domain is extended accordingly. Furthermore, the notion of concrete features is introduced. These are features which can be built using aggregation over roles followed by features. Then \( ALC(D + \Sigma) \)-concepts are defined.

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

The set of concrete features is inductively defined as follows:

- Each feature name \( f \in N_F \) is a concrete feature,
- feature chains are concrete features,
- if \( R \in N_R \) is a role, \( f \) is a concrete feature, and \( \Sigma \in \text{agg}(D) \) is an aggregation function, then \( \Sigma(Rf) \) is a concrete feature.

Finally, \( ALC(D + \Sigma) \)-concepts are obtained from \( ALC(D) \)-concepts by allowing, additionally, the use of concrete features \( f_i \) in predicate restrictions \( P(f_1, \ldots, f_n) \) (recall that in \( ALC(D) \), only feature chains were allowed).

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

**Definition 6** An \( ALC(D + \Sigma) \)-interpretation \( I \) is an \( ALC(D) \)-interpretation \( I \) which additionally satisfies

\[
(S(Rf))^I = \{(x, y) \in \Delta^2 \times \text{dom}(D) \mid \Sigma^D(M^\text{Ref}(f))^I = y \}
\]

where, for \( x \in \Delta^2 \), a concrete feature \( f \), and a role \( R \), \( M^\text{Ref}(f) \) denotes the multiset over \( \text{dom}(D) \) where the number of occurrences of \( z \in \text{dom}(D) \) is determined by the number of \( R^2 \)-successors \( y \) of \( x \) with \( f^2(y) = z \), i.e., for \( z \in \text{dom}(D) \) we have

\[
M^\text{Ref}(f)(z) := \# \{ y \in \Delta^2 \mid (x, y) \in R^2 \text{ and } f^2(y) = z \}.
\]

We point out two consequences of this definition, which might not be obvious at first sight: (a) if \((R \circ f)^I(x)\) contains individuals in \( \Delta^2 \), then these individuals have no influence on \( M^\text{Ref}(f) \): it is defined in such a way that it takes only into account \((R \circ f)^I\)-successors of \( x \) in the concrete domain \( \text{dom}(D) \); (b) if \( M^\text{Ref}(f) \) is not finite, then the outcome depends on \( D \) and \( \Sigma \); for example, the minimum of a (possibly infinite) subset of the positive integers is always defined, whereas the sum is undefined for infinite subsets.

Unfortunately, the following theorem shows that admissibility of a concrete domain does not longer guarantee decidability of the interesting inference problems:

**Theorem 7** For a concrete domain \( D \) where

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

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

**Remarks:** (a) The aggregation functions \( \min, \max, \text{sum} \) are supposed to be defined as usual, i.e., “\( \geq \)” is an extension of “\( \geq \)” on \( \mathbb{N} \), and

\[
\text{sum}(M) = \begin{cases} 
\sum_{y \in M} M(y) \cdot y & \text{if } M \text{ is finite} \\
\text{undefined} & \text{otherwise}
\end{cases}
\]

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

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

(b) At first sight, this undecidability result may seem to be rather restricted. Note, however, that it just requires that \( \text{dom}(D) \) contains the non-negative integers. Furthermore, the aggregation functions \( \min, \max, \text{sum} \) are among those normally considered as built-in functions in databases (see, for example, [Gupta et al. 1995; Muknick & Shmuel 1995; Levy & Muknick 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.

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

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

does not exist in \( \mathbb{N}^m \). When building the reduction concept \( C_{P, Q} \), one encounters three major problems: (a) We
only know that \( \text{dom}(\mathcal{D}) \) 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) The reduction asks for the simulation of calculations such as addition, multiplication, and exponents. 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 \( C_{P,Q} \).

In the following, we sketch how these problems can be solved—details and the definition of \( C_{P,Q} \) can be found in [Baader & Sattler 1997]: (a) is solved by making use of the concept
\[
E^R_2 := (\forall R. (P_{=1}(f)) \cap P_{=} (\text{sum}(R \circ f), g)),
\]
whose instances have as \( g \)-successor the number of their \( R \)-successors. Hence their \( g \)-successor is in \( \mathbb{N} \) or undefined (if there are infinitely many \( R \)-successors). (b) Addition can be realized by the aggregation function \( \text{sum} \) and multiplication (and hence exponentiation) can be reduced to addition. (c) This problem is solved by introducing features \( x_i \) for each variable \( x_i \) and by making strong use of the concept \( \text{inv} \). All \( R \)-successors of an instance \( a \) of \( \text{inv} \) have the same \( x_i \)-successor, which equals the \( x_i \)-successor of \( a \).

This concept can be used to guarantee that all “relevant” individuals in a model of \( C_{P,Q} \) have the same \( x_i \)-successor for each variable \( x_i \).

Then 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. \( Q \)). Each leaf of these trees stands for one of the variables \( x_i \), whose value is “spread” over the whole structure using the concept \( \text{inv} \) described above.

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 \( \Sigma \), \( f \) is a feature name (and not a complex feature chain or concrete feature).

A closer investigation of the concept \( C_{P,Q} \) reveals that
(a) negation does not occur, (b) no concept of the form \( \exists R.C \) is used, and (c) the only place where disjunction \( \cup \) occurs is in concepts \( E^R_n \) describing individuals having exactly \( n \) \( R \)-successors (which are used to represent the coefficients of the polynomials):
\[
E^R_n := \forall R. \left( \bigcup_{1 \leq i \leq n} P_{=1}(f_i) \right) \cap \forall R. \left( \bigcap_{1 \leq i \leq n} (P_{=1}(f_i) \Rightarrow (\bigcap_{j \neq i} \perp(f_j))) \right) \cap \\
\bigcap_{1 \leq i \leq n} P_{=} (\text{sum}(R \circ f_i)).
\]

For an instance \( a \) of \( E^R_n \), every \( R \)-successor has an \( f_i \)-successor for exactly one \( i \), \( 1 \leq i \leq n \), and this \( f_i \)-successor has value 1 (first two lines). The constraint on the concrete feature \( \text{sum}(R \circ f_i) \) (third 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. In \( A\mathcal{C}(D + \Sigma) \), with \( D \) as described in the preconditions of Theorem 7, it seems to be impossible to describe the fact that an individual has exactly \( n \) \( R \)-successors without using union. However, given a concrete domain \( D \) that provides, in addition to what was required in Theorem 7, for all non-negative integers \( n \) a unary predicate \( P_{=n} \) that tests for equality with \( n \), then the following concept \( E^R_n' \) can be used to describe those individuals having exactly \( n \) \( R \)-successors:
\[
E^R_n' := \forall R. P_{=1}(f) \cap P_{=} (\text{sum}(R \circ f)).
\]

Hence, the reduction concept \( C_{P,Q} \) can be rewritten using only conjunction \( \cap \) and universal value restriction \( \forall R.C \). As introduced in [Baader 1999], let \( \mathcal{F}_C \) denote the set of those concepts that are built using conjunction and universal value restriction only, and let \( \mathcal{F}_C(D + \Sigma) \) denote the extension of this language by concrete domains and aggregates. Then the following corollary is an immediate consequence of the remarks made above.

**Corollary 8** For a concrete domain \( D \) where
- \( \text{dom}(D) \) includes the non-negative integers \( \mathbb{N} \),
- \( \text{pred}(D) \) contains, for all non-negative integers \( n \), (unary) predicates \( P_{=n} \) that test for equality with \( n \), and the (binary) equality predicate \( P_{=} \),
- \( \text{agg}(D) \) contains \( \min, \max, \text{sum} \),

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

Undecidability of satisfiability is shown, as sketched above, by a reduction of Hilbert’s 10th problem. From this, undecidability of subsumption follows because a concept \( C \) is satisfiable iff it is not subsumed by an unsatisfiable concept, and because the \( \mathcal{F}_C(D + \Sigma) \)-concept \( C_{\perp} := \top(f) \cap \bot(f) \) 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 grows that are processed by these systems, 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 min, max, sum that suffice to obviate undecidability are the most “well-behaved” ones: aggregation functions like count or average are much more difficult to handle. For example, min, max, sum are monotonic, i.e., if $S \subseteq S'$, then:

\[
\begin{align*}
\min(S) & \geq \min(S'), \\
\max(S) & \leq \max(S'), \\
\sum(S) & \leq \sum(S').
\end{align*}
\]

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

Arguing from another perspective, $\mathcal{ALC}(D+\Sigma)$ is a rather expressive Description Logic and it might not be very surprising that adding aggregation to $\mathcal{ALC}(D)$ 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 $\mathcal{E}_{\mathcal{F}C_1(x)}$ constructed from $C_1$ is in the answer set of a conjunctive query $\pi_{C_2}$ constructed from $C_2$. This reduction is, for several reasons, not possible for $\mathcal{FL}_0(D+\Sigma)$-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


