D W Q

Foundations of **D**ata **W**arehouse **Q**uality

National Technical University of Athens (NTUA)
Informatik V & Lehr- und Forschungsgebiet Theoretische Informatik (RWTH)
Institute National de Recherche en Informatique et en Automatique (INRIA)
Deutsche Forschungszentrum für künstliche Intelligenz (DFKI)
University of Rome «La Sapienza» (Uniroma)
Istituto per la Ricerca Scientifica e Tecnologica (IRST)

F. Baader, U. Sattler

Description Logics with Concrete Domains and Aggregation

Proceedings of the 13th European Conference on Artificial Intelligence (ECAI '98),

John Wiley & Sons Ltd, 1998.

DWQ : ESPRIT Long Term Research Project, No 22469
Contact Person : Prof. Yannis Vassiliou, National Technical University of Athens,
15773 Zographou, GREECE Tel +30-1-772-2526 FAX: +30-1-772-2527, e-mail: yv@cs.ntua.gr

Description Logics with Concrete Domains and Aggregation

Franz Baader [1] and **Ulrike Sattler** [1]

Abstract. We extend different Description Logics by concrete domains (such as integers and reals) and by aggregation functions over these domains (such as $\mathsf{min}, \mathsf{max}, \mathsf{count}, \mathsf{sum}$), which are usually available in database systems. On the one hand, we show that this extension may lead to undecidability of the basic inference problems satisfiability and subsumption. This is true even for a very small Description Logic and very simple aggregation functions, provided that universal value restrictions are present. On the other hand, disallowing universal value restrictions yields decidability of satisfiability, 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 $\mathsf{min}, \mathsf{max}, \mathsf{count}$.

1 Motivation

Unlike many other expressive representation formalisms, such as database schema and query languages, basic Description Logic formalisms (e.g., $\mathcal{ALC}$ [12]) do not allow for built-in predicates (like comparisons of numbers) and for aggregation functions (like sum, min, max, $\mathsf{average}$, count). The first deficit was overcome in [3], where a generic extension of $\mathcal{ALC}$ by a *concrete domain* $\mathcal{D}$ was proposed. In this extended Description Logic, 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

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

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

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

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`.

In Section 2, we present a generic extension of $\mathcal{ALC}(\mathcal{D})$ by aggregation that is based on the idea mentioned above of introducing new "aggregated features". Unfortunately, it turns out that, given a concrete domain together with aggregation functions satisfying some rather weak conditions, this extension has an undecidable satisfiability and subsumption 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 aggregation already causes undecidability; see Section 3.

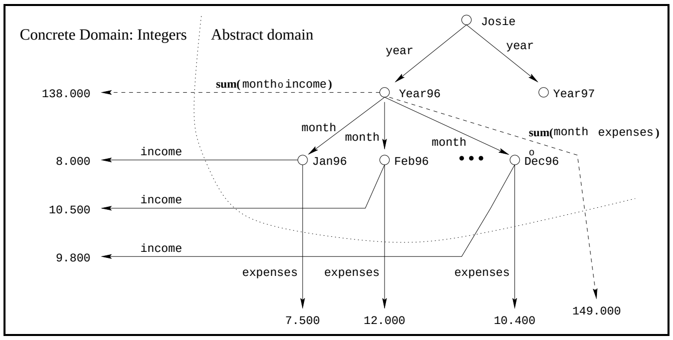

Figure 1. Example for aggregation

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 restrictions and restricting negation to concept names, is defined. In Section 4, we present a tableau-based algorithm that decides satisfiability of $\mathcal{CQ}(\mathcal{D}_{\mathsf{agg}})$-concepts, provided that satisfiability of finite conjunctions of predicates involving aggregations on multiset variables in the concrete domain $\mathcal{D}$ can be decided. An example of such a concrete domain is the set of (nonnegative) integers with comparisons ($=$, $\leq$, $\leq_n$, ...) and the aggregation functions $\mathsf{min}, \mathsf{max}, \mathsf{count}$.

Full proofs of all results presented can be found in [1; 2].

2 Syntax and semantics of $\mathcal{ALC}(\mathcal{D}_{\mathsf{agg}})$

The Description Logic $\mathcal{ALC}(\mathcal{D}_{\mathsf{agg}})$, which will be introduced in this section, is based on $\mathcal{ALC}(\mathcal{D})$ [3], which extends the well-known Description Logic $\mathcal{ALC}$ [12] by a so-called *concrete domain*. In principle, a concrete domain consist of a set (e.g., of numbers) together with predicates on this set. In our Description Logic $\mathcal{ALC}(\mathcal{D}_{\mathsf{agg}})$, we additionally assume that the concrete domain is equipped with aggregation functions. In order to define aggregation functions appropriately, we must introduce the notion of a *multiset*: in contrast to simple sets, in a multiset an individual may occur more than once; for example, the multiset $\{\{1\}\}$ is different from the multiset $\{\{1, 1\}\}$. Multisets are necessary to ensure, e.g., that `Josie`'s income is also calculated correctly if she earns the same amount of money in more than one month.

Definition 1 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 over 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

[1] RWTH-Aachen, LuFG Theoretical Computer Science, Ahornstr. 55, 52074 Aachen, Germany. This work was supported by the Esprit Project 22469 – DWQ.

ECAI 98. *13th European Conference on Artificial Intelligence*
Edited by Henri Prade
Published in 1998 by John Wiley & Sons, Ltd.

enumerate the members a_i of a finite multiset, we use the notation $\{\{a_1, \ldots, a_n\}\}$ to distinguish multisets from sets.

Since the aggregation functions strongly depend on the specific concrete domains, they are directly included in the definition of a concrete domain.

Definition 2 A *concrete domain* $\mathcal{D}$ consists of

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

Each predicate symbol $P \in \mathsf{pred}(\mathcal{D})$ is associated with an arity n and an n-ary relation $P^{\mathcal{D}} \subseteq \mathsf{dom}(\mathcal{D})^n$, and each aggregation function $\Sigma \in \mathsf{agg}(\mathcal{D})$ is associated with a partial function $\Sigma^{\mathcal{D}}$ from the set of multisets over $\mathsf{dom}(\mathcal{D})$ into $\mathsf{dom}(\mathcal{D})$.

In [3], concrete domains do not contain aggregation functions, and they are restricted to so-called *admissible* concrete domains in order to keep the inference problems of $\mathcal{ALC}(\mathcal{D})$ 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. Since no aggregation functions are involved, these conjunctions contain only concrete predicates applied to variables standing for *individuals* in $\mathsf{dom}(\mathcal{D})$.

Aggregation functions are used to build *concrete features* out of roles and features. The syntax of these concrete features and of $\mathcal{ALC}(\mathcal{D}_{\mathsf{agg}})$-concepts is defined as follows:

Definition 3 Let N_C, N_R, N_F be disjoint sets of *concept*, *role*, and *feature names*, 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 $f_1 \ldots f_n\, \Sigma(R \circ f)$ is a concrete feature.

The set of $\mathcal{ALC}(\mathcal{D}_{\mathsf{agg}})$*-concepts* is the smallest set such that

- every concept name is a concept,
- 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 concrete features, 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 expressions, their semantics is defined in the usual model-theoretic way.

Definition 4 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 $\cdot^{\mathcal{I}}$ 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 $f^{\mathcal{I}} : \Delta^{\mathcal{I}} \to \Delta^{\mathcal{I}} \cup \mathsf{dom}(\mathcal{D})$. 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.

The interpretation $\mathcal{I}$ is now extended to concepts and concrete features as follows:

$$\begin{aligned}
(C \sqcap D)^{\mathcal{I}} &= C^{\mathcal{I}} \cap D^{\mathcal{I}},\ (C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}, \neg C^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}},\\
(\exists R.C)^{\mathcal{I}} &= \{d \in \Delta^{\mathcal{I}} \mid \exists e \in \Delta^{\mathcal{I}} : (d, e) \in R^{\mathcal{I}} \wedge e \in C^{\mathcal{I}}\},\\
(\forall R.C)^{\mathcal{I}} &= \{d \in \Delta^{\mathcal{I}} \mid \forall e \in \Delta^{\mathcal{I}} : (d, e) \in R^{\mathcal{I}} \Rightarrow e \in C^{\mathcal{I}}\},\\
P(u_1, \ldots, u_n)^{\mathcal{I}} &= \{x \in \Delta^{\mathcal{I}} \mid (u_1^{\mathcal{I}}(x), \ldots, u_n^{\mathcal{I}}(x)) \in P^{\mathcal{D}}\},\\
(f_1 \ldots f_m)^{\mathcal{I}}(x) &= f_m^{\mathcal{I}}(f_{m-1}^{\mathcal{I}}(\ldots (f_1^{\mathcal{I}}(x) \ldots),\\
(\Sigma(R \circ f))^{\mathcal{I}}(a) &= \begin{cases} \Sigma^{\mathcal{D}}(M_a^{(R \circ f)^{\mathcal{I}}}) & \text{if } M_a^{(R \circ f)^{\mathcal{I}}} \text{ is a multiset,}\\ \text{undefined} & \text{otherwise.}\end{cases}\\
(u\Sigma(R \circ f))^{\mathcal{I}}(x) &= \Sigma(R \circ f)^{\mathcal{I}}(u^{\mathcal{I}}(x))
\end{aligned}$$

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 $\mathcal{I}$. An individual $x \in C^{\mathcal{I}}$ is called an *instance* of a concept C.

We point out three consequences of the above definition, which might not be obvious at first sight:

(a) By definition, an instance of the concept $P(u_1, \ldots, u_n)$ has an $u_i^{\mathcal{I}}$-successor in $\mathsf{dom}(\mathcal{D})$ for each $i, 1 \leq i \leq n$.

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

(c) There are two reasons for $(\Sigma(R \circ f))^{\mathcal{I}}(a)$ to be undefined. 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 is not defined.

3 The undecidability results

The results presented in [3] imply that subsumption and satisfiability are decidable for $\mathcal{ALC}(\mathcal{D})$-concepts, provided that $\mathcal{D}$ is admissible. For $\mathcal{ALC}(\mathcal{D}_{\mathsf{agg}})$, admissibility of the concrete domain does no longer guarantee decidability of these inference problems:

Theorem 1 *For a concrete domain $\mathcal{D}$ where*

- $\mathsf{dom}(\mathcal{D})$ *includes the nonnegative 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* $\mathsf{min}, \mathsf{max}, \mathsf{sum}$,[2]

satisfiability and subsumption of $\mathcal{ALC}(\mathcal{D}_{\mathsf{agg}})$-concepts are undecidable.

This undecidability result is rather strong for the following reasons: (a) It does not require that $\mathsf{dom}(\mathcal{D})$ is the set of nonnegative integers, but only that it *contains* the nonnegative integers. This is an important generalisation since, for example, the real numbers have better computational properties than the nonnegative integers: the first-order theory of $+, \cdot, \leq$ is undecidable over the nonnegative integers, whereas it is decidable over the reals.

(b) The aggregation functions $\mathsf{min}, \mathsf{max}, \mathsf{sum}$ are among the simplest normally considered as built-in functions in databases (see, e.g., [7; 9; 8; 13]). In addition, to test whether a certain value equals 1 or whether two values are equal is possible in all database systems with built-in predicates.

[2] Restricted to the nonnegative integers, these aggregation functions are assumed to be defined as usual.

ECAI 98. *13th European Conference on Artificial Intelligence*
Edited by Henri Prade
Published in 1998 by John Wiley & Sons, Ltd.

(c) There are admissible concrete domains (in the sense of [3]) that satisfy the preconditions of Theorem 1. Examples are the (nonnegative) integers, reals, and rationals together with the predicates $P_{=1}, P_{=}, P_{\neq}, P_{\neq 1}$.

Sketch of the proof of Theorem 1: The proof is by reduction of Hilbert's 10th problem [5] to satisfiability of concepts, i.e., for given polynomials $P, Q \in \mathbb{N}[x_1, \ldots, x_m]$, we construct an $\mathcal{ALC}(\mathcal{D}_{\mathsf{agg}})$-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 $\mathbb{N}^m$. Undecidability of subsumption follows because C is satisfiable iff C is not subsumed by $\bot$, where $\bot$ denotes an arbitrary trivially unsatisfiable concept.

The idea underlying the reduction is to represent the (sub)term structure of the polynomial P (resp. Q) as a tree, which is related to an instance a of $C_{P,Q}$ via the feature P (resp. Q). To this purpose, for each variable x_i, a feature $\mathtt{x}_i$ is introduced. Then we simulate the calculations in both subterm trees using aggregation functions. To enforce a solution of Equation 1, we enforce that the value of $P(x_1, \ldots, x_m)$ (which is represented as the $P \circ s$-successor of a) equals the value of $Q(x_1, \ldots, x_m)$ (represented as $Q \circ s$-successor of a).

Since a complete description of the reduction concept is too complex to be presented here, we just highlight two of the problems that we had to overcome when defining the reduction:

(a) Hilbert's 10th problem asks for a solution in $\mathbb{N}^m$, whereas $\mathsf{dom}(\mathcal{D})$ is only required to *contain* $\mathbb{N}$. To enforce that the $\mathtt{x}_i$-successors (which stand for the values assigned to the variables x_i) are nonnegative integers, we use the concept

$$E^R_{\mathtt{x}_i} := (\forall R.(P_{=1}(f))) \sqcap P_{=}(\mathsf{sum}(R \circ f), \mathtt{x}_i).$$

An instance of $E^R_{\mathtt{x}_i}$ has as its $\mathtt{x}_i$-successor the number of its R-successors. Note that our definition of the semantics of predicate restrictions implies that the $\mathtt{x}_i$-successor is defined, and that the definition of the semantics of aggregated features requires that this number is finite, i.e., an element of $\mathbb{N}$.

(b) The features $\mathtt{x}_i$ occur at various positions in the trees representing the polynomials. To ensure that all individuals in both trees have the same $\mathtt{x}_i$-successors (i.e., the variable x_i is evaluated by the same number everywhere), we make strong use of the concept Inv:

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

where $\top(\mathtt{x}_i)$ is an abbreviation for $P_{=}(\mathtt{x}_i, \mathtt{x}_i)$, which simply ensures that there exists an $\mathtt{x}_i$-successor. Inv is defined such that all 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. It is used to propagate the value of x_i to all parts of the trees.

The necessary calculations (addition, multiplication, and exponentiation) are realised by first expressing addition using the aggregation function sum, and by reducing multiplication and exponentiation to addition.

A closer investigation of the concept $C_{P,Q}$ used for the reduction reveals that it does not require the full expressive power of $\mathcal{ALC}(\mathcal{D}_{\mathsf{agg}})$. In the full paper [1] it is shown that one can dispense with negation, existential value restrictions ($\exists R.C$), and disjunction in the definition of the reduction concept, provided that the concrete domain satisfies a slightly stronger property. To be more precise, let $\mathcal{FL}_0$ denote the set of concepts that can be built using conjunction and universal value restriction ($\forall R.C$) only, and let $\mathcal{FL}_0(\mathcal{D}_{\mathsf{agg}})$ denote the extension of this Description Logic by a concrete domain with aggregation.

Theorem 2 *For a concrete domain $\mathcal{D}$ where*

- $\mathsf{dom}(\mathcal{D})$ *includes the nonnegative integers* $\mathbb{N}$,
- $\mathsf{pred}(\mathcal{D})$ *contains, for all nonnegative 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 $\mathcal{FL}_0(\mathcal{D}_{\mathsf{agg}})$-concepts is undecidable.

4 The decidability result

The undecidability results presented above strongly depend on the presence of universal value restrictions. In order to obtain a decidable Description Logic with aggregation, we remove universal value restrictions from the set of constructors. Since negation together with existential value restrictions would re-introduce universal restrictions, we must also restrict the use of negation. To be more precise, the Description Logic $\mathcal{CQ}(\mathcal{D}_{\mathsf{agg}})$ is obtained from $\mathcal{ALC}(\mathcal{D}_{\mathsf{agg}})$ by disallowing universal value restrictions and by allowing negation to occur only in front of concept names. Due to these restrictions, we can design a tableau-based algorithm that decides satisfiability of $\mathcal{CQ}(\mathcal{D}_{\mathsf{agg}})$-concepts, provided that the concrete domain satisfies some additional restrictions. Given an input concept C, it tries to construct a model of C by breaking down C, thus making explicit all constraints imposed by C. In a first phase, it checks for "abstract" consistency, whereas the second phase checks the satisfiability of all concrete predicates imposed on individuals in this model. One difference to the algorithm in [3] is that these predicates may involve aggregated multiset variables besides variables for individuals in $\mathsf{dom}(\Delta^{\mathcal{I}})$. The data structure this algorithm works on are constraints.

Definition 5 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 $\mathsf{agg}(\mathcal{D})(\sigma)$. Constraints are of the form:

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

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 $f_1 \ldots f_n\, \Sigma(R \circ f)$ of a in S iff there is an $f_1 \ldots f_n$-successor b of a in S and $(b, Y) : (R \circ f) \in S$.

Definition 6 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:

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

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

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

ECAI 98. *13th European Conference on Artificial Intelligence*
Edited by Henri Prade
Published in 1998 by John Wiley & Sons, Ltd.

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

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

is satisfiable in $\mathcal{D}$, where $x \in \tau_{\mathcal{D}}$ are variables for elements in $\mathsf{dom}(\mathcal{D})$ and $Y \in \sigma$ 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 $x \in \tau_{\mathcal{D}}$ and $b \in \tau_A$.

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).

Note that a fork $\{(x,\ell) : f, (x,\ell') : f\}$ (resp. $\{(a,Y) : (R \circ f), (a,Z) : (R\circ f)\}$) implies $\ell^{\mathcal{I}} = \ell'^{\mathcal{I}}$ (resp. $Y^{\mathcal{I}} = Z^{\mathcal{I}}$) for all models $\mathcal{I}$ of S. This is made explicit by fork elimination. If x has both a concrete and an abstract f-successor, then this is an obvious inconsistency because the concrete domain is disjoint from the abstract one and leads therefor to a clash.

The tableau-based *completion algorithm* for deciding satisfiability of $\mathcal{CQ}(\mathcal{D}_{\mathsf{agg}})$-concepts applies *completion rules* given 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 $S \rightarrow S_i$, 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.

A remark concerning Rule 3.b. is in order. In contrast to rules dealing with existential restrictions employed by other tableau-based algorithms, which always introduce a new individual, this rule considers different alternatives: either a new individual is introduced or an already existing R-successor is re-used. This nondeterministic variant is necessary for the correct treatment of aggregation functions whose value depends on the multiplicity of elements: this is, e.g., the case for count and sum, but not for min and max. If *new* R-successors were generated for each existential restriction, satisfiability of the following concept would not be detected because it would always lead to constraint systems that are not $\mathcal{D}$-consistent (since the multiset of all $R\circ f$-successors would be too large).

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

If the concrete domain is restricted to aggregation functions whose outcome does not depend on the multiplicity of elements in the input multiset, then Rule 3.b. can be substituted by a deterministic one that generates, for each existential restriction, a new R-successor.

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

2. Disjunction: if $a : (C_1 \sqcup C_2) \in S$ and $a : C_1 \notin S$ and $a : C_2 \notin S$, then
$S \rightarrow \quad S_1 = S \cup \{a : C_1\}$,
$S \rightarrow \quad S_2 = S \cup \{a : C_2\}$.

3.a. Existential feature restriction: 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 role restriction: If $a : (\exists R.C) \in S$ for a role name R, $\{b_1, \ldots, b_n\}$ are all R-successors of a, and for all i, $1 \leq i \leq n$, $b_i : C \notin S$, then
$S \rightarrow \quad S_i = S \cup \{b_i : C\}$,
$S \rightarrow \quad S_{n+1} = S \cup \{(a,b) : R, b : C\}$,
for a new variable $b \in \tau_A$.

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

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

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
$S \rightarrow \quad S \cup \{P(\alpha_1, \ldots, \alpha_n)\} \cup \bigcup_{1 \leq i \leq n} S_i$.
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
$S \rightarrow \quad S \cup \{z : Y\}$.

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

As a consequence of the nondeterministic Rule 3.b., we may restrict our attention to models $\mathcal{I}$ of the constraint system in which different role successors in the system are interpreted by distinct individuals in the model.

Definition 7 An *m-model* $\mathcal{I}$ of a $\mathcal{CQ}(\mathcal{D}_{\mathsf{agg}})$-constraint system S is a model that satisfies $b^{\mathcal{I}} \neq c^{\mathcal{I}}$ for all $b, c \in \tau_A$ with $\{(a,b) : R, (a,c) : R\} \subseteq S$ for some $a \in \tau_A$ and $R \in N_R$.

Lemma 1 below implies that decidability of satisfiability of $\mathcal{CQ}(\mathcal{D}_{\mathsf{agg}})$-concepts is reduced by the completion algorithm to decidability of $\mathcal{D}$-consistency. In the spirit of [3], concrete domains $\mathcal{D}$ where $\mathcal{D}$-consistency is decidable are called *m-admissible*. Examples of m-admissible concrete domains with min, max, and count as aggregation functions will be presented below.

Lemma 1 *Let C_0 be a $\mathcal{CQ}(\mathcal{D}_{\mathsf{agg}})$-concept and let S be a constraint system obtained by applying the completion rules to $\{x_0 : C_0\}$. Then*

- *if C_0 is satisfiable, then $\{x_0 : C_0\}$ has an m-model.*

ECAI 98. *13th European Conference on Artificial Intelligence*
Edited by Henri Prade
Published in 1998 by John Wiley & Sons, Ltd.

- *for each completion rule $\mathcal{R}$ that can be applied to S, and for each interpretation $\mathcal{I}$ we have that $\mathcal{I}$ is an m-model of S iff $\mathcal{I}$ is an m-model of one of the systems S_i obtained by applying $\mathcal{R}$.*
- *if S is a complete, $\mathcal{D}$-consistent, and clash-free constraint system, then S has an m-model.*
- *if S contains a clash or is not $\mathcal{D}$-consistent, then S does not have an m-model.*
- *the completion algorithm terminates when applied to $\{x_0 : C_0\}$.*

It should be noted that the fourth part of the lemma would not hold if "m-model" were replaced by "model". It is easy to see that the concept $(*)$ yields an example: the case where two distinct R-successors are generated leads to a constraint system which has a model and is not $\mathcal{D}$-consistent. The third part of the lemma is the point where the absence of universal value restrictions becomes important.

Theorem 3 *If $\mathcal{D}$ is an m-admissible concrete domain, then satisfiability of $\mathcal{CQ}(\mathcal{D}_{\mathsf{agg}})$-concepts is decidable.*

Indeed, after the completion algorithm has terminated, it has generated a finite tree whose leaves are all labelled with complete constraint systems. As an immediate consequence of Lemma 1, the input concept C_0 is satisfiable iff one of these complete systems is clash-free and $\mathcal{D}$-consistent.

The following lemma provides examples of nontrivial m-admissible concrete domains.

Lemma 2 *The following conditions imply m-admissibility of the concrete domain $\mathcal{D}$:*

- $\mathsf{dom}(\mathcal{D})$ *is the set of nonnegative integers, integers, rational numbers or reals,*
- $\mathsf{pred}(\mathcal{D}) = \{<, \leq, >, \geq, =,\} \cup \bigcup_{n \in \mathsf{dom}(\mathcal{D})} \{<_n, \leq_n, >_n, \geq_n, =_n\}$, *where the equalities and inequalities are interpreted in the usual way, and $\{<_n, \leq_n, >_n, \geq_n, =_n\}$ are unary predicates testing for being less than n, less or equal n, etc., and*
- $\mathsf{agg}(\mathcal{D}) = \{\mathsf{min}, \mathsf{max}, \mathsf{count}\}$.

The proof of this lemma applies the techniques introduced in [11]: aggregated multiset variables of the form $\mathsf{max}(Y), \mathsf{min}(Y), \mathsf{count}(Y)$ are replaced by new individual variables $y_{\mathsf{max}}, y_{\mathsf{min}}, y_{\mathsf{count}}$. In addition, one must add formulae that appropriately axiomatise the behaviour of the aggregation functions. For example, $y_{\mathsf{min}} \leq y_{\mathsf{max}}$ must hold, and if we know that z belongs to Y (because the constraint system contains $z : Y$), then $y_{\mathsf{min}} \leq z \leq y_{\mathsf{max}}$ must hold as well. Similarly, y_{count} must be at least as large as the number of individual variables known to belong to Y (plus possibly 1 or 2, depending on whether the the minimum/maximum coincides with one of these variables or not). This reduction yields a formula built from equalities and inequalities between individual variables, which can easily be tested for satisfiability.

5 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 [11; 7; 9; 8; 13]. The undecidability results are orthogonal to those presented in [9] since our prerequisites are weaker and no recursion mechanisms are used. Neither are they implied by the undecidability results in [11]: 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 [10] for containment of conjunctive queries with aggregation functions 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 Description Logic $\mathcal{CQ}(\mathcal{D}_{\mathsf{agg}})$ and conjunctive queries with aggregation functions is a topic for future research. Finally, the extensions of Description Logics with aggregation presented in [4; 6] cannot be compared to those introduced here because their inference algorithms do not take into account concrete predicates or aggregation functions.

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 functions. Thus, it is important to find additional concrete domains with aggregation functions for which $\mathcal{D}$-consistency is decidable. For example, what happens if the aggregation function count in Lemma 2 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 [11].

© 1998 F. Baader and U. Sattler
ECAI 98. *13th European Conference on Artificial Intelligence*
Edited by Henri Prade
Published in 1998 by John Wiley & Sons, Ltd.

REFERENCES

[1] F. Baader and U. Sattler, 'Description logics with aggregates and concrete domains', Technical Report 97-01, LuFg Theoretical Computer Science, RWTH Aachen, (1997). Available via www: http://www-lti.informatik.rwth-aachen.de/Forschung/Papers.html.

[2] F. Baader and U. Sattler, 'Description logics with aggregates and concrete domains, part II', Technical Report 98-02, LuFg Theoretical Computer Science, RWTH Aachen, (1998). Available via www: http://www-lti.informatik.rwth-aachen.de/Forschung/Papers.html.

[3] Franz Baader and Philipp Hanschke, 'A schema for integrating concrete domains into concept languages', in *Proc. of IJCAI-91*, pp. 452–457, Sydney, (1991).

[4] Ti. Catarci, G. D'Angiolini, and M. Lenzerini, 'Conceptual language for statistical data modeling', *Data and Knowledge Engineering*, **17**(2), 93–125, (1995).

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

[6] G. De Giacomo and P. Naggar, 'Conceptual data model with structured objects for statistical databases', in *Proc. of SSDBM'96*, pp. 168–175. IEEE Computer Society Press, (1996).

[7] A. Gupta, V. Harinarayan, and D. Quass, 'Aggregate-query processing in data warehousing environments', in *Proc. of VLDB-95*, (1995).

[8] A. Y. Levy and I. S. Mumick, 'Reasoning with aggregation constraints', in *Proc. of EDBT-96*, Avignon, France, (1996).

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

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

[11] K.A. Ross, D. Srivastava, P. J. Stuckey, and S. Sudarshan, 'Foundations of aggregation constraints', *Theoretical Computer Science*, (1998). To appear.

[12] Manfred Schmidt-Schauß and Gert Smolka, 'Attributive concept descriptions with complements', *Artificial Intelligence*, **48**(1), 1–26, (1991).

[13] D. Srivastava, S. Dar, H. V. Jagadish, and A. Y. Levy, 'Answering queries with aggregation using views', in *Proc. of VLDB-96*, Bombay, India, (1996).