A Knowledge-Based Approach to Visual Information[*]

Elisa Bertino[*] Ahmed K. Elmagarmid[†] Mohand-Saïd Hacid[†]

[*]Dipartimento di Scienze dell'Informazione
University of Milano
Via Comelico, 39/41 20135 Milano - Italy
bertino@dsi.unimi.it

[†]Department of Computer Sciences
Purdue University
West Lafayette, IN 47907 - USA
{*ake,mshacid*}*@cs.purdue.edu*

Abstract

We propose an approach based on description logics for the representation and retrieval of visual information. We first consider objects as having shapes which are described by means of semi-algebraic sets[1]*. We propose a model which consists of three layers: (1)* Shape Layer*, which provides the geometric shapes of image objects; (2)* Object Layer*, intended to contain objects of interest and their description; and (3)* Schema Layer*, which contains the structured abstractions of objects, i.e., a general schema about the classes of objects represented in the* Object Layer*. We propose two abstract languages on the basis of description logics: one for describing knowledge of the object and schema layers, and the other, more expressive, for making queries. Queries can refer to the form dimension (i.e., information of the* Shape Layer*) or to the semantic dimension (i.e., information of the* Object Layer*). These languages employ a variable free notation. Second, we show how this framework can be easily extended to accommodate the visual layer (e.g., color and texture). Queries in this framework may be time-consuming, and resorting to the use of materialized*[2] *views to process and optimize such queries may be useful. For that, we propose a formal framework for testing containment of a query in a view expressed in our query language.*

Keywords : *Multimedia Databases, Shape Representation, Content-based Access, Query Containment, Query Optimization, Semi-algebraic Sets, Reasoning, Description Logics.*

1 Introduction

Interacting with visual content is essential to visual information retrieval [6]. New tools and interaction paradigms should permit the searching for visual data by referring to both visual content and textual descriptions. There are two essential questions associated with

[*]This work is supported by the National Science Foundation under grants 9972883-EIA, 9974255-IIS and 9983249-EIA; grants from HP, IBM, Intel, Telcordia and CERIAS; and Indiana 21st Century grant from the State of Indiana.

[1]i.e., equalities and inequalities between integer polynomials in several indeterminates. That is, a relational approach as opposed to feature-vector approach and shape through transformation approach.

[2]A materialized view is a query whose a physical copy of each instance, answer to the query, is stored and maintained.

content-based query systems for visual data: (1) How to specify queries, and (2) How to access the intended data efficiently for given queries. These queries may be formulated in terms of a number of different features, and can be grossly classified into three categories [20]: (1) *form queries*, addressing visual objects on the basis of color, texture, sketch, or shape specifications; (2) *content queries*, focusing on domain concepts, spatial constraints, or various types of attributes; (3) *mixed queries*, which combine the two previous categories. In order to deal with these questions, formal representations of information to enable users and query optimizers to take explicit advantage of the nature of image data are required.

Possible visual elements are color, texture and shapes. However, among these features, shape is the most important because it represents significant regions or relevant objects (in images for example). Ideally, shape segmentation would be automatic and efficient, but it is either impossible or difficult and human intervention is needed to give a few orientations. After obtaining relevant objects, suited representations must be chosen among multiple existing models. In the sequel, we first consider shape representation and retrieving objects by making reference, in queries, to that component. Then, we extend our framework to accommodate the other visual features (i.e., color and texture).

We take a new look at the problem of modeling and querying visual data and find that knowledge representation and reasoning techniques for concept languages developed in Artificial Intelligence, appropriately extended, provide an interesting angle to attack such problems. These techniques also provide a nice basis for semantic query optimization in visual databases. We exploit the possibility of using two languages: one for defining the schema (i.e. the structure) of a database and populating it, and the other, more expressive, for querying the database through the schema. These languages are equipped with *sound*, *complete*, and *terminating* inference procedures, that allow various forms of reasoning to be carried out on the intensional level of the database. We believe that the use of a formal language for describing visual data can provide several benefits. First of all, it semantically clarifies the process of visual information retrieval. Secondly, it allows to express complex queries to a visual database. Third, it allows a compilation of recognition[3] (e.g., images can be classified along known categories as soon as they are entered in the database).

The contributions of this paper are developed in a step-wise fashion, as follows: First, based on the scheme developed in [3] we show how to handle semi-algebraic sets in description logics. Semi-algebraic sets are used to represent geometric shapes associated with objects. Second, we show that this framework can easily accommodate the content based access of visual objects (e.g., by color or texture). Third, we show that query containment in our query language is decidable. Containment[4] of queries is the problem of checking whether the result of one query is contained in what another query produces [1, 26]. Containment is mainly concerned with query optimization.

Paper outline: In Section 2, we develop our languages and give their Tarski-style extensional semantics. Section 3 explains how the visual aspect (i.e., color and texture) of objects can be taken into account in our framework. Section 4 provides a logical calculus for query containment. Section 5 discusses related work. We conclude in Section 6.

[3]Mainly with a description logic based language.

[4]Also called implication.

2 The Languages

2.1 Preliminaries

As the representational formalisms presented in the following belong to the family of description logics, we first briefly introduce these logics. Description logics (also called concept logics, terminological logics, or concept languages) [9, 21, 7] are a family of logics designed to represent the taxonomic and conceptual knowledge of a particular application domain on an abstract, logical level. They are equipped with well-defined, set-theoretic semantics. Furthermore, the interesting reasoning problems such as subsumption and satisfiability are, for most description logics, decidable (see, for example, [15]).
Starting from atomic concepts and roles[5], complex concepts (and roles) are built by using a set of constructors. For example, from atomic concepts `Human` and `Female` and the atomic role `child` we can build the expression $\texttt{Human} \sqcap \forall\texttt{child}.\texttt{Female}$ which denotes the set of all `Human` whose children are (all) instances of `Female`. Here, the symbol $\sqcap$ denotes conjunction of concepts, while $\forall$ denotes (universal) value restriction.
A knowledge base in a description logic system is made up of two components: (1) the *TBox* is a general schema concerning the classes of individuals to be represented, their general properties and mutual relationships; (2) the *ABox* contains a partial description of a particular situation, possibly using the concepts defined in the *TBox*. It contains descriptions of (some) individuals of the situation, their properties and their interrelationships.
Retrieving information in a knowledge base system based on description logics is a deductive process involving both the schema (*TBox*) and its instantiation (*ABox*). In fact, the *TBox* is not just a set of constraints on possible *ABoxes*, but contains intensional information about classes. This information is taken into account when answering queries to the knowledge base. The following reasoning services are the most important ones provided by knowledge representation systems based on description logics (See [16] for an overview):

- *Concept satisfiability*: Given a knowledge base and a concept C, does there exist at least one model of the knowledge base assigning a non-empty extension to C?
- *Subsumption*: Given a knowledge base and two concepts C and D, is C more general than D? That is, is each instance of D also an instance of C in all models of the knowledge base?
- *Knowledge base satisfiability*: Are an *ABox* and a *TBox* consistent with each other? That is, does the knowledge base admit a model?
- *Instance checking*: Given a knowledge base, an individual o, its (partial) description, and a concept C, is o an instance of C in any model of the knowledge base?

Various database applications for which these reasoning services are useful are mentioned, e.g., in [2, 10].

In the following, we are interested in concept satisfiability and subsumption. An *unsatisfiable* query is suggestive of an *empty answer*. A query containment problem will be reduced to a subsumption problem for concepts described in an appropriate description logic.

Before we give the syntax and semantics of our *abstract languages*, we introduce some useful definitions.

[5]A concept is interpreted as a class of objects in the domain of interest, and then can be considered as an unary predicate. Roles are interpreted as binary relations on individuals, and then considered as binary predicates.

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

- *the domain* $\mathsf{dom}(\mathcal{D})$,
- *a set of predicate symbols* $\mathsf{pred}(\mathcal{D})$*, where 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 many applications (in particular when querying databases), one would like to be able to refer to concrete domains and predicates on these domains when defining queries. An example of such a concrete domain could be the set of (nonnegative) integers with comparisons $(=, <, \leq, \geq, >)$.

Concrete domains are used to incorporate application-specific domains (i.e., strings, integers, reals, etc.) into the abstract domain of individuals (objects). In [3], concrete domains are restricted to so-called *admissible* concrete domains in order to keep the inference problems[6] decidable. We recall that, roughly spoken, a concrete domain $\mathcal{D}$ is called *admissible* iff (1) $\mathsf{pred}(\mathcal{D})$ is closed under negation and contains a unary predicate name $\top_{\mathcal{D}}$ for $\mathsf{dom}(\mathcal{D})$, and (2) satisfiability of finite conjunctions over $\mathsf{pred}(\mathcal{D})$ is decidable. For example, semi-algebraic sets defined by a finite number of polynomial equations or inequalities as defined in [4] are admissible concrete domains.

Definition 2 (Real Formula) *A real formula is a well-formed first-order logic formula built from equalities and inequalities between integer polynomials in several indeterminates, i.e.,*

- *if* p *is a polynomial with real coefficients over the variables* $x_1, \ldots, x_m$ *over the real numbers, then* $p(x_1, \ldots, x_m)\ \theta\ 0$ *is a real formula with* $\theta \in \{=, <, \leq, >, \geq, \neq\}$*;*
- *if* φ *and* ψ *are real formulas, then so are* $\varphi \wedge \psi$*,* $\varphi \vee \psi$*, and* $\neg\varphi$*; and*
- *if* x *is a real variable and* φ *is a real formula in which* x *occurs free, then* $\exists x \varphi(x)$ *is a real formula.*

If φ is a real formula with m free variables $x_1, \ldots, x_m$, then it defines a subset $\varphi_{\mathbb{R}^m}$ of the m-dimensional Euclidean space $\mathbb{R}^m$, namely the set of points satisfying φ:

$$\varphi_{\mathbb{R}^m} := \{(u_1, \ldots, u_m) \in \mathbb{R}^m \mid \varphi(u_1, \ldots, u_m)\}.$$

Definition 3 (Semi-algebraic sets [4]) *Point-sets defined by real formulas are called semi-algebraic sets.*

For example, the set $V = \{(x, y) \in \mathbb{R}^2 \mid x^2 + y^2 = 1\}$ is semi-algebraic.
Note that semi-algebraic sets of $\mathbb{R}^m$ make the smallest family of subsets of $\mathbb{R}^m$ such that:

1. it contains all the sets of the form $\{(u_1, \ldots, u_m) \in \mathbb{R}^m \mid P(u_1, \ldots, u_m) \geq 0\}$;
2. it is closed with respect to the set-theoretic operations of finite union, finite intersection and complementation.

In the following, we denote the domain of semi-algebraic sets, i.e., dom(semi-algebraic sets), by $\top_{\mathcal{SAS}}$. Furthermore, it is considered as a particular concrete domain.

Every real formula can effectively be transformed into a quantifier-free real formula [25, 14]. As a consequence, it is *decidable* whether a real sentence is *valid* or *satisfiable* in the ordered

[6]Such as subsumption, instantiation, and consistency.

field of the real numbers.

Figure 1 shows a part of geographical information about intercity highways and railroads. The names are fictitious. In this example, linear constraints (see tables 1 , 2 and 3), which can be seen as linear restrictions of the polynomial constraints are used to specify spatial representations of the different objects (i.e., *cities*, *highways*, and *railroads*). Dotted lines represent intercity railroads while plain lines represent intercity highways. The positions of the objects are given in a 2-dimensional space in the map co-ordinate system. In this example, the position of all objects is described by *linear* (dis)equations, hence this simple formalism is expressive enough for the representation of cities and transportation ways. Simple theoretically, it is interesting from a practical point of view.

The linear data model is particularly suited to model spatial data in applications in which geometrical information is required and in which this information can be approximated by linear geometrical spatial objects. This model is opposed to the topological one which is suitable for applications in which rather than exact geometrical information, the relative position of spatial objects is of importance. With regard to the expressive power and complexity of linear constraint query languages, see [18, 27].

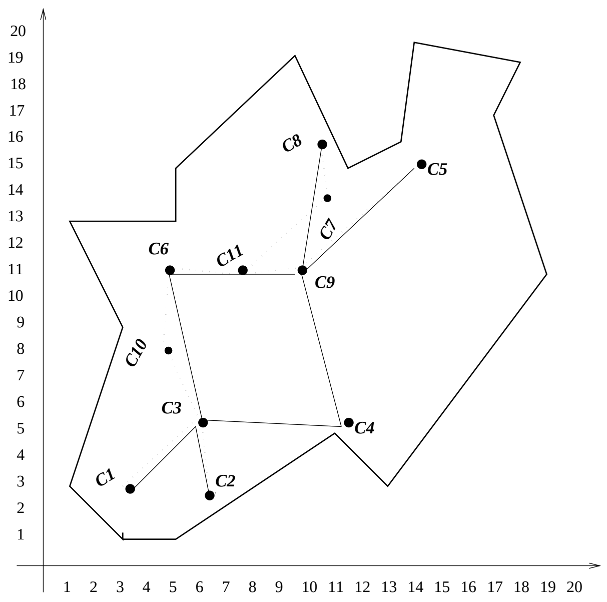

Figure 1: Example of a geographical multimedia information about intercity highways and railroads

Table 1: City

Oid	Name	Population	Geometry
id_1	$C1$	250.000	$(x = 3.5) \wedge (y = 2.5)$
id_2	$C2$	290.800	$(x = 6.4) \wedge (y = 2.5)$
id_3	$C3$	1.004.928	$(x = 6) \wedge (y = 5.3)$
...	...	...	...

Table 2: Highway

Oid	**Name**	**Geometry**
id_1'	H-$C1$-$C3$	$(y \leq 5.3) \wedge (y \geq 2.5) \wedge$ $(140x - 110y \leq 201)$
id_2'	H-$C2$-$C3$	$(y \leq 5.3) \wedge (y \geq 2.5) \wedge$ $(140x + 10y \leq 893)$
...	...	...

Table 3: Railroad

Oid	**Name**	**Geometry**
id_1''	R-$C1$-$C3$	$(y \leq 5.3) \wedge (y \geq 2.5) \wedge$ $(140x - 110y \leq 201)$
id_2''	R-$C2$-$C3$	$(y \leq 5.3) \wedge (y \geq 2.5) \wedge$ $(140x + 10y \leq 893)$
...	...	...

2.2 The Schema Language ($\mathcal{SL}$)

We now introduce a simple description logic that will be used for describing the schema part of a database. Starting from atomic concepts and roles, complex concepts are built by using the universal quantification ($\forall$) and the existential quantification ($\exists$) constructors.

The syntax and the semantics of this description logic are given below.

Definition 4 (Syntax) *Let N_C, N_R, N_f be three pairwise disjoint sets of concept names, role names, and feature (i.e., functional role) names respectively, $\mathcal{D}_1, \ldots, \mathcal{D}_k$ be concrete domains, and $\top_{\mathcal{SAS}}$ be the set of semi-algebraic sets. Let P be a role name, f be a feature name, A be a concept name, A' be a concept name or a concrete domain name, and A'' be a concrete domain name or the universal concept ($\top$). Concept terms C, D are defined by the following rules:*

$$
\begin{array}{lll}
C, D & \longrightarrow & A \mid & \textit{(primitive concept)} \\
 & & \forall P.A \mid & \textit{(typing of role)} \\
 & & \forall f.A' \mid & \textit{(typing of feature)} \\
 & & \exists P.\top \mid & \textit{(necessary role)} \\
 & & \exists f.A'' \mid & \textit{(necessary feature)} \\
 & & \exists f.\top_{\mathcal{SAS}} & \textit{(necessary spatial feature)}
\end{array}
$$

Let A, A_1, and A_2 be concept names, A_3 be a concept name, a concrete domain name or $\top_{\mathcal{SAS}}$, D be a concept term, P be a role name, and f be a feature name. Then $A \dot{\preceq} D$ (and we say A is a $\mathrm{sub-concept}$ of D), $P \dot{\preceq} A_1 \times A_2$, $f \dot{\preceq} A_1 \times A_3$ are called *axioms*.

$A \dot{\preceq} D$ is called a primitive concept specification, where D gives necessary conditions for membership in A. The axioms $P \dot{\preceq} A_1 \times A_2, f \dot{\preceq} A_1 \times A_3$ give the definition of the role P and the feature f respectively. Domain and range of a role or a feature are restricted by concepts or concrete domains.

A $\mathcal{SL}$ schema $\mathcal{S}$ consists of a finite set of axioms.

Figure 2 shows a fragment of a database schema. Concrete domains needed here are `STRING`, `INTEGER`, and $\top_{\mathcal{SAS}}$.

$Country \dot{\preceq} \forall continent.Continent$
$Country \dot{\preceq} \forall population.\texttt{INTEGER}$
$Country \dot{\preceq} \forall political\_situation.\texttt{STRING}$
$Country \dot{\preceq} \forall area.\texttt{INTEGER}$
$Country \dot{\preceq} \forall average\_summer\_temperature.\texttt{INTEGER}$

$City \dot{\preceq} \forall name.\texttt{STRING}$
$City \dot{\preceq} \forall in\_country.Country$
$City \dot{\preceq} \forall accommodation.Accommodation$
$City \dot{\preceq} \exists location.\top_{\mathcal{SAS}}$
......

$Highway \dot{\preceq} \exists name.\texttt{STRING}$
$Highway \dot{\preceq} \exists spatial\_description.\top_{\mathcal{SAS}}$

$Railroad \dot{\preceq} \exists name.\texttt{STRING}$
$Railroad \dot{\preceq} \exists spatial\_description.\top_{\mathcal{SAS}}$

$River \dot{\preceq} \exists name.\texttt{STRING}$
$River \dot{\preceq} \exists spatial\_description.\top_{\mathcal{SAS}}$
......

Figure 2: A fragment of a geographical database schema

Definition 5 (Semantics) *The semantics is given by an interpretation $\mathcal{J} = (\Delta^{\mathcal{J}}, \cdot^{\mathcal{J}})$, which consists of an (abstract) interpretation domain $\Delta^{\mathcal{J}}$, and an interpretation function $\cdot^{\mathcal{J}}$. The abstract domain has to be disjoint from any given concrete domain, i.e., $\Delta^{\mathcal{J}} \cap \mathsf{dom}(\mathcal{D}_i) = \emptyset$ for all concrete domain $\mathcal{D}_i$ ($i \in [1,k]$), the concrete domains are pairwise disjoint, and $\mathsf{pred}(\mathcal{D}_i) \cap \mathsf{pred}(\mathcal{D}_j) = \emptyset$ for $i \neq j$. The interpretation function $\cdot^{\mathcal{J}}$ associates each concept C with a subset $C^{\mathcal{J}}$ of $\Delta^{\mathcal{J}}$, each role P with a binary relation $P^{\mathcal{J}}$ on $\Delta^{\mathcal{J}}$, and each feature name f with a partial function $f^{\mathcal{J}} : \Delta^{\mathcal{J}} \to (\Delta^{\mathcal{J}} \cup (\bigcup_{i=1}^{k} \mathsf{dom}(\mathcal{D}_i)) \cup \top_{\mathcal{SAS}})$. Additionally, $\mathcal{J}$ has to satisfy the following equations:*

$$\begin{array}{rcl}
(\forall P.A)^{\mathcal{J}} & = & \{d \in \Delta^{\mathcal{J}} \mid \forall d' \\
 & & \qquad (d^{\mathcal{J}}, d'^{\mathcal{J}}) \in P^{\mathcal{J}} \to d'^{\mathcal{J}} \in A^{\mathcal{J}}\} \\
(\forall f.A')^{\mathcal{J}} & = & \{d \in \Delta^{\mathcal{J}} \mid \text{ if } f^{\mathcal{J}}(d^{\mathcal{J}}) \text{ is defined then} \\
 & & \qquad f^{\mathcal{J}}(d^{\mathcal{J}}) \in A'^{\mathcal{J}}\} \\
(\exists f.A'')^{\mathcal{J}} & = & \{d \in \Delta^{\mathcal{J}} \mid f^{\mathcal{J}}(d^{\mathcal{J}}) \text{ is defined and} \\
 & & \qquad f^{\mathcal{J}}(d^{\mathcal{J}}) \in A''^{\mathcal{J}}\} \\
(\exists P.\top)^{\mathcal{J}} & = & \{d \in \Delta^{\mathcal{J}} \mid \exists d' : (d^{\mathcal{J}}, d'^{\mathcal{J}}) \in P^{\mathcal{J}}\} \\
(\exists f.\top)^{\mathcal{J}} & = & \{d \in \Delta^{\mathcal{J}} \mid \exists d' : f^{\mathcal{J}}(d^{\mathcal{J}}) = d'^{\mathcal{J}}\} \\
(\exists f.\top_{\mathcal{SAS}})^{\mathcal{J}} & = & \{d \in \Delta^{\mathcal{J}} \mid f^{\mathcal{J}}(d^{\mathcal{J}}) \text{ is defined as} \\
 & & \text{a satisfiable real formula}\}
\end{array}$$

An interpretation $\mathcal{J}$ satisfies the axiom $A \dot{\preceq} D$ iff $A^{\mathcal{J}} \subseteq D^{\mathcal{J}}$, the axiom $P \dot{\preceq} A_1 \times A_2$ iff $P^{\mathcal{J}} \subseteq A_1^{\mathcal{J}} \times A_2^{\mathcal{J}}$, and the axiom $f \dot{\preceq} A_1 \times A_3$ iff $f^{\mathcal{J}} \subseteq A_1^{\mathcal{J}} \times A_3^{\mathcal{J}}$. Here, $A_3^{\mathcal{J}}$ stands for the domain of A_3 (i.e., $\mathsf{dom}(A_3)$) for all $\mathcal{J}$.
In the following: **(1)** individuals of the abstract domain are called *abstract individuals*, and those of a concrete domain are called *concrete individuals*; **(2)** we assume the *Unique Name Assumption* for abstract individuals but not for concrete individuals. If we want to treat a unique name assumption for the concrete individuals we have to require that the concrete domain contains a predicate name equality.

Definition 6 (Model) *An interpretation $\mathcal{J} = (\Delta^{\mathcal{J}}, .^{\mathcal{J}})$ is a model, also called a valid interpretation, of a schema $\mathcal{S}$ iff it satisfies every axiom in $\mathcal{S}$.*

An interpretation $\mathcal{J}$ that satisfies all axioms in $\mathcal{S}$ is called an $\mathcal{S}$-interpretation.

The language introduced previously allows to describe knowledge about classes of individuals and relationships between these classes. We can now turn our attention to the extensional level, which we call the *ABox*. The *ABox* essentially allows one to specify instance-of relations between individuals and concepts, and between pairs of individuals and roles or features.

Definition 7 *Let N_I and N_D be two disjoint alphabets of symbols, called abstract individual names and concrete individual names respectively. Instance-of relationships are expressed in terms of* membership assertions *of the form:*

$$a : C,\ \ (a, b) : P,\ \ (a, b) : f,\ \ (a, z) : f,\ \ (z_1, \ldots, z_n) : P_r$$

*where a and b are abstract individual names, $z, z_1, \ldots, z_n$ are concrete individual names, C is a concept name or an arbitrary concept, P is a role name, P_r is an n-ary predicate name of a concrete domain, and f is a feature name. Intuitively, the first form states that a is an instance of C, and the second form states that a is related to b by means of the role P (we also say b is a P-*successor *of a).*

In order to assign a meaning to membership assertions, the extension function $.^{\mathcal{I}}$ of an interpretation $\mathcal{I}$ is extended to individuals by mapping them to elements of $\Delta^{\mathcal{I}}$ in such a way that $a^{\mathcal{I}} \neq b^{\mathcal{I}}$ if $a \neq b$ (Unique Name Assumption). *For concrete individuals, the unique name assumption does not hold.*
An interpretation $\mathcal{I}$ satisfies the assertion:

$$a : C \text{ iff } a^{\mathcal{I}} \in C^{\mathcal{I}}, \quad (a, b) : P \text{ iff } (a^{\mathcal{J}}, b^{\mathcal{J}}) \in P^{\mathcal{J}}, (a, b) : f \text{ iff } f^{\mathcal{I}}(a^{\mathcal{I}}) = b^{\mathcal{I}}$$

$$(a, z) : f \text{ iff } f^{\mathcal{I}}(a^{\mathcal{I}}) = z^{\mathcal{I}}, (z_1, \ldots, z_n) : P_r \text{ iff } (z_1^{\mathcal{I}}, \ldots, z_n^{\mathcal{I}}) \in P_r^{\mathcal{D}}$$

An *ABox* $\mathcal{A}$ is a finite set of membership assertions.
An interpretation $\mathcal{I}$ is a model for an *ABox* $\mathcal{A}$ iff $\mathcal{I}$ satisfies all the assertions in $\mathcal{A}$.

For example, the following facts can be considered as a fragment of an image database maintained by a traveling agency which sells stays on resorts:

$\mathsf{paris} : \mathsf{City}$, $(\mathsf{paris}, \mathsf{N}_1) : \mathsf{name}$, $\mathsf{N}_1 :=_{\text{"PARIS"}}$, $(\mathsf{paris}, \mathsf{P}) : \mathsf{population}$,
$\mathsf{P} :=_{10000000}$, $(\mathsf{paris}, \mathsf{I}) : \mathsf{image}$, $\mathsf{I} :=_{\mathsf{picture_{p01}}}$, $\mathsf{eiffel\_tower} : \mathsf{Monument}$, $(\mathsf{eiffel\_tower}, \mathsf{N}_2) : \mathsf{name}$,
$\mathsf{N}_2 :=_{\text{"Tour Eiffel"}}$, $(\mathsf{eiffel\_tower}, \mathsf{Z}) : \mathsf{location}$,
$(Z) : 2 \leq X \leq 17 \wedge 22 \leq Y \leq 53, \ldots$

The first statement says that paris is an individual, instance of the concept City. The last two assertions say that eiffel_tour has a location in a space (defined by a map's co-ordinate system) given by $2 \leq X \leq 17 \wedge 22 \leq Y \leq 53$.

2.3 The Query Language ($\mathcal{QL}$)

Querying a database means retrieving stored objects that satisfy certain *conditions* and hence are interesting for a user. In the case of relational databases, queries are constructed by means of algebra expressions defined on relations from the database. As a property, answers are also relations (i.e., sets of tuples). This correspondence between database entities and answer formats presents advantages that lead to the design and development of query optimization techniques. In object-oriented databases, classes are used to represent sets of objects. By analogy with the relational approach, classes can be used for describing query results. If such a possibility exists, then we can consider some kind of *reasoning* on the structure[7] of classes that will lead to reveal, for example, *containment* relationships between queries.

[7]And hence the semantics of class hierarchies.

In our framework, we follow this approach. *Queries are represented as concepts* in our abstract language.
In the following, we give the syntax and semantics of a concept language for making queries.

Definition 8 (Syntax) *Let A be a concept name, P be an atomic role, a be an abstract individual name, $u, u_1, u_1^{'}, \ldots$ be feature chains*[8], *$P_r \in \mathsf{pred}(\mathcal{D}_i)$ for some $i \in [1,k]$ be an n-ary predicate name (called ordinary predicate name), P_s be a binary spatial predicate name, P_{os_i} for $i \in [1,m]$ be spatial or ordinary binary predicate names, and Φ be a real formula. Concepts C, D and roles R can be formed by means of the following syntax:*

$$C, D \longrightarrow \top \mid \{a\} \mid A \mid C \sqcap D \mid \exists R.C \mid P_r(u_1, \ldots, u_n) \mid \\ P_s^{(\Phi)}(u) \mid \Theta(C, D, \{\langle u_1, P_{os_1}, u_1^{'}\rangle, \ldots, \langle u_m, P_{os_m}, u_m^{'}\rangle\})$$

$$R \longrightarrow P \mid P^{-} \mid f \mid f^{-}$$

For example, the following simple query finds all cities located in the area $(x \leq 8) \wedge (x \geq 2) \wedge (y \leq 6) \wedge (y \geq 1)$ which are crossed by a river.

$$in^{((x\leq 8)\wedge(x\geq 2)\wedge(y\leq 6)\wedge(y\geq 1))}(location) \sqcap \\ \Theta(City, River, \{\langle location, \mathsf{intersect}, spatial\_description\rangle\})$$

In this example, the binary predicate name intersect stands for the spatial intersection between two shapes.

Definition 9 (Semantics) *The semantics is given by an interpretation $\mathcal{J} = (\Delta^{\mathcal{J}}, \cdot^{\mathcal{J}})$, which consists of an (abstract) interpretation domain $\Delta^{\mathcal{J}}$, and an interpretation function $\cdot^{\mathcal{J}}$. The abstract domain has to be disjoint from any given concrete domain, i.e., $\Delta^{\mathcal{J}} \cap \mathsf{dom}(\mathcal{D}_i) = \emptyset$ for all concrete domain $\mathcal{D}_i$ ($i \in [1,k]$), the concrete domains are pairwise disjoint, and $\mathsf{pred}(\mathcal{D}_i) \cap \mathsf{pred}(\mathcal{D}_j) = \emptyset$, for $i \neq j$. The interpretation function $\cdot^{\mathcal{J}}$ associates each concept C with a subset $C^{\mathcal{J}}$ of $\Delta^{\mathcal{J}}$, each role P with a binary relation $P^{\mathcal{J}}$ on $\Delta^{\mathcal{J}}$, and each feature name f with a partial function $f^{\mathcal{J}} : \Delta^{\mathcal{J}} \rightarrow (\Delta^{\mathcal{J}} \cup (\bigcup_{i=1}^{k} \mathsf{dom}(\mathcal{D}_i)) \cup \top_{\mathcal{SAS}})$. Additionally, $\mathcal{J}$ has to satisfy the following equations:*

$$\begin{array}{rcl}
\top^{\mathcal{J}} & = & \Delta^{\mathcal{J}} \\
\{a\}^{\mathcal{J}} & = & \{a^{\mathcal{J}}\} \\
(C \sqcap D)^{\mathcal{J}} & = & C^{\mathcal{J}} \cap D^{\mathcal{J}} \\
(\exists R.C)^{\mathcal{J}} & = & \{d \in \Delta^{\mathcal{J}} \mid \exists d' : (d^{\mathcal{J}}, d'^{\mathcal{J}}) \in R^{\mathcal{J}} \wedge \\
 & & \qquad d'^{\mathcal{J}} \in C^{\mathcal{J}}\} \\
P_r(u_1, \ldots, u_n)^{\mathcal{J}} & = & \{d \in \Delta^{\mathcal{J}} \mid \exists z_1, \ldots, z_n \in \mathsf{dom}(\mathcal{D}) \\
 & & u_1^{\mathcal{J}}(d^{\mathcal{J}}) = z_1^{\mathcal{J}}, \ldots, u_n^{\mathcal{J}}(d^{\mathcal{J}}) = z_n^{\mathcal{J}} \text{ and} \\
 & & (z_1^{\mathcal{J}}, \ldots, z_n^{\mathcal{J}}) \in P_r^{\mathcal{D}}\} \\
P_s^{(\Phi)}(u)^{\mathcal{J}} & = & \{d \in \Delta^{\mathcal{J}} \mid u^{\mathcal{J}}(d^{\mathcal{J}}) \text{ satisfies } P_s^{(\Phi)} \text{ in } \top_{\mathcal{SAS}}\}
\end{array}$$

$$\begin{array}{l}
(\Theta(C, D, \{\langle u_1, P_{os_1}, u_1^{'}\rangle, \ldots, \langle u_m, P_{os_m}, u_m^{'}\rangle\}))^{\mathcal{J}} = \\
\qquad \{d \in \Delta^{\mathcal{J}} \mid d \in C^{\mathcal{J}} \text{ and } \exists d',\ d' \in D^{\mathcal{J}} \text{ such that} \\
\qquad (u_1^{\mathcal{J}}(d), u_1^{'\mathcal{J}}(d')) \text{ satisfies } P_{os_1} \text{ in } \mathcal{D}_{P_{os_1}} \text{ and } \ldots \text{ and} \\
\qquad (u_m^{\mathcal{J}}(d), u_m^{'\mathcal{J}}(d')) \text{ satisfies } P_{os_m} \text{ in } \mathcal{D}_{P_{os_m}}\}
\end{array}$$

3 Extension to the Visual Layer

So far, we have presented both a data model for specifying structural associations and spatial representation of objects and a query language allowing to retrieve objects based on these

[8]A feature chain is a composition of features.

structural associations and spatial relationships. The result of a query is a set of objects identifiers (those satisfying the conditions of the query). In this section, we show how to consider the visual aspect. For example, a query may include a condition concerning the color of objects to be retrieved. Color is a visual feature which is immediately perceived when looking at a visual object. Retrieval by color similarity requires that models of color stimuli are used, such that distances in the color space correspond to human perceptual distances between colors.

The extension to visual features can be done through the use of the interface between an abstract domain and a concrete domain defined previously. The visual layer of objects can be seen as a concrete domain whose elements are visual objects and whose predicates are similar-to predicates.

As an example, consider a traveling agency which sells stays on resorts. This agency has a database containing both textual information and images (compact representations) about resorts (such as cities, art galleries, lodging, etc.). Before selling a traveling to a customer, he/she is invited to virtually discover his tour by referring to information contained in the database. Figure 3 shows a simple fragment of the structure of such a database. Each inclusion assertion (introduced by $\dot{\preceq}$) imposes a constraint on the instances of the class it refers to. The concrete domains required here are INTEGER, STRING, and IMAGE. The domain IMAGE is a set of images structured by a set of predicates (e.g., similar-to predicates).

$Country \dot{\preceq} \forall continent.Continent$
$Country \dot{\preceq} \forall population.\mathtt{INTEGER}$
$Country \dot{\preceq} \forall political\_situation.\mathtt{STRING}$
$Country \dot{\preceq} \forall area.\mathtt{INTEGER}$
$Country \dot{\preceq} \forall image.\mathtt{IMAGE}$
$Country \dot{\preceq} \forall ast.\mathtt{INTEGER}$

$City \dot{\preceq} \forall name.\mathtt{STRING}$
$City \dot{\preceq} \forall in\_country.Country$
$City \dot{\preceq} \forall accommodation.Accommodation$
$City \dot{\preceq} \forall image.\mathtt{IMAGE}$

$Hotel \dot{\preceq} Accommodation$
$Hotel \dot{\preceq} \forall price\_single.\mathtt{INTEGER}$
$Hotel \dot{\preceq} \forall price\_double.\mathtt{INTEGER}$
$Hotel \dot{\preceq} \forall image.\mathtt{IMAGE}$

$Room \dot{\preceq} Accommodation$
$Room \dot{\preceq} \forall price.\mathtt{INTEGER}$
...

$Camping \dot{\preceq} Accommodation$
$Camping \dot{\preceq} \forall price.\mathtt{INTEGER}$
$Camping \dot{\preceq} \forall image.\mathtt{IMAGE}$
$Camping \dot{\preceq} \forall in\_city.City$

$Site \dot{\preceq} \forall name.\mathtt{STRING}$

$Monument \dot{\preceq} Site$
$Monument \dot{\preceq} \forall opening\_hour.\mathtt{INTEGER}$
$Monument \dot{\preceq} \forall closing\_hour.\mathtt{INTEGER}$
$Monument \dot{\preceq} \forall opening\_days.Day$
$Monument \dot{\preceq} \forall image.\mathtt{IMAGE}$
$Monument \dot{\preceq} \forall price.\mathtt{INTEGER}$

$Amusement\_parc \dot{\preceq} Site$
$Amusement\_parc \dot{\preceq} \forall price.\mathtt{INTEGER}$
$Amusement\_parc \dot{\preceq} \forall image.\mathtt{IMAGE}$

$Art\_galery \dot{\preceq} Site$
$Art\_galery \dot{\preceq} \forall price.\mathtt{INTEGER}$

$Beach \dot{\preceq} \forall image.\mathtt{IMAGE}$
$Beach \dot{\preceq} \forall awt.\mathtt{INTEGER}$

...

Figure 3: An image database structure

Given the schema of figure 3, the query:

$$\begin{aligned}&\Theta(\mathsf{Camping}, \{\mathsf{d}_{\mathsf{example}}\},\\ &\quad \{\langle \mathsf{image}, \mathsf{same-texture}, \mathsf{image}\rangle, \langle \mathsf{image}, \mathsf{same-color}, \mathsf{image}\rangle\}) \sqcap\\ &\qquad \exists \mathsf{in\_city}.(=_{\text{"Germany"}} (\mathsf{name}) \sqcap <_{100} (\mathsf{price}))\end{aligned}$$

would be "Find a set of camping in Germany, with a price below 100, and the picture (i.e., the filler for the feature image) of each camping object in the answer set is similar to the picture associated with the individual $d_{example}$ regarding texture and color". Here image is an attribute which links an abstract individual to a picture in the Feature & Content Layer. The predicates same-texture and same-color belong to this layer.

4 A Calculus for Deciding Query Containment

In this section we provide a calculus for deciding the containment of a query in a view (which is a query as well). In particular we present the calculus and state its correctness and completeness.
A query Q is $\mathcal{S}$-satisfiable if there is an $\mathcal{S}$-interpretation $\mathcal{J}$ such that $Q^{\mathcal{J}} \neq \emptyset$. We say that a query Q is $\mathcal{S}$-subsumed by a view V (or Q is $\mathcal{S}$-contained in a view V) (written $Q \dot{\preceq}_{\mathcal{S}} V$) if $Q^{\mathcal{J}} \subseteq V^{\mathcal{J}}$ for every $\mathcal{S}$-interpretation $\mathcal{J}$.

Definition 10 (Containment) *Given a $\mathcal{SL}$ schema $\mathcal{S}$, a query Q and a view V in the $\mathcal{QL}$ language, are the answers to Q also answers to V for any database state obeying the schema $\mathcal{S}$.*

The basic idea for deciding the *containment* of a query Q in a view V is drawn from [11]. We take an object o and transform Q into a prototypical database state where o is an answer to Q. We do so by generating individuals, entering them into concepts in the schema, and relating them through roles and features. If o belongs to the answer of V, then Q is contained in V. If not, we have a state where an individual is in the answer to Q but not in the answer to V and therefore V does not contain Q.

In the following, we make three assumptions:

- The schema $\mathcal{S}$ is acyclic.
- Given a concept name A and a role (or a feature) name R, we do not allow axioms of the form $A \dot{\preceq} \exists R.A', A \dot{\preceq} \exists R.A''$ in the schema $\mathcal{S}$, where A' and A'' are two different concept names.
- We do not allow sub-expressions of the form $\exists R.C_1 \sqcap \ldots \sqcap \exists R.C_n$ for the same role (or feature) name R in a query, but we allow sub-expressions of the form $\exists R.(C_1 \sqcap \ldots \sqcap C_n)$.

4.1 Propagation Rules

According to the syntax of our concept languages, concepts describing queries make reference (through roles and features) to abstract individual names and/or concrete individual names. In the following, a and b are abstract individual names, x, y are variables denoting abstract individuals. The only difference between variables and abstract individual names is that for variables, the *unique name assumption* does not hold, hence two different variables can be interpreted as the same individual—in contrast to what was said for abstract individual names. In the sequel, we refer to abstract individual names and variables as abstract individuals, denoted by the letters $s, t, s', t', s_1, t_1, \ldots$. The unique name assumption does not hold for concrete individual names and we use $z, z', z_1, z_1^{'}, z_2, z_2^{'}, \ldots$ as names for concrete individuals. Finally, we use $v, v', v_1, v_1^{'}, \ldots$ to refer to abstract individual names, variables, or concrete individual names.

For the sake of simplicity, we do not distinguish between individual names denoting arbitrary concrete individuals (i.e., `INTEGER`*,* `STRING`*, etc.) and individual names denoting real* *formulas (from $\top_{\mathcal{SAS}}$).*

The calculus works on syntactic entities called *constraints* which are of the form:

$s : C$, $(s,t) : R$, $(s,t) : f$, $(s,z) : f$, $(z_1, \ldots, z_n) : P_r$, $(z) : P_s^{(\Phi)}$, $(z_1, z_2) : P_{os}$, $a \neq b$

where s, t are abstract individuals, a, b are abstract individual names, $z, z_1, \ldots, z_n$ are concrete individual names, R is a role name, f is a feature name, P_r is an ordinary n-ary predicate name, $P_s^{(\Phi)}$ is an unary spatial predicate name, P_{os} is an ordinary binary predicate name or a spatial binary predicate name, and C is a $\mathcal{QL}$ concept or a $\mathcal{SL}$ concept name. Intuitively, the first form states that s is an instance of C, and the second form states that s is related to t by means of the role R (we also say t is a R-successor of s).

A *constraint system* $\tilde{S}$ is a finite set of constraints.

The semantics is extended to constraints. An interpretation $\mathcal{J}$ maps a variable x to an element $x^{\mathcal{J}}$ of the abstract domain $\Delta^{\mathcal{J}}$ of $\mathcal{J}$ and a concrete individual name z to an element of its concrete domain. An interpretation $\mathcal{J}$ satisfies a constraint

$s : C$ iff $s^{\mathcal{J}} \in C^{\mathcal{J}}$, $(s,t) : R$ iff $(s^{\mathcal{J}}, t^{\mathcal{J}}) \in R^{\mathcal{J}}$,
$(s,t) : f$ iff $f^{\mathcal{J}}(s^{\mathcal{J}}) = t^{\mathcal{J}}$, $a \neq b$ iff $a \neq b$,
$(s,z) : f$ iff $f^{\mathcal{J}}(s^{\mathcal{J}}) = z^{\mathcal{J}}$,
$(z_1, \ldots, z_n) : P_r$ iff $(z_1^{\mathcal{J}}, \ldots, z_n^{\mathcal{J}}) \in P_r^{\mathcal{D}}$,
$(z) : P_s^{(\Phi)}$ iff $z^{\mathcal{J}}$ satisfies $P_s^{(\Phi)}$ in $\top_{\mathcal{SAS}}$,
$(z_1, z_2) : P_{os}$ iff $(z_1^{\mathcal{J}}, z_2^{\mathcal{J}})$ satisfies P_{os} in $\mathcal{D}_{P_{os}}$

A constraint system $\tilde{S}$ is *satisfiable* if there is an interpretation $\mathcal{J}$ that satisfies every constraint in $\tilde{S}$.

Let c and c' be constraints and $\mathcal{S}$ be a schema. we write

$$c \models_{\mathcal{S}} c'$$

if every $\mathcal{S}$-model of c is also an $\mathcal{S}$-model of c'.

Proposition 1 *Let $\mathcal{S}$ be a schema, Q be a query and V be a view, and x' be a variable. Then*

$$Q \dot{\preceq}_{\mathcal{S}} V \text{ iff } x' : Q \models_{\mathcal{S}} x' : V$$

Proof See Appendix.

Hence, to test Q and V for *containment*, we have to check the constraints $x' : Q$ and $x' : V$ for *entailment.*

Let V be a view. We call the constraint $x' : V$ a goal.
As in [11], our method makes use of four kinds of rules: *decomposition*, *schema*, *goal*, and *composition* rules. Given a query Q and a view V, the rules work on pairs of constraint systems $\mathcal{Q}.\mathcal{V}$. We call $\mathcal{Q}$ (built from Q) the *facts* and $\mathcal{V}$ (built from V) the *goal.* To decide whether $Q \dot{\preceq}_{\mathcal{S}} V$, we take a variable x' and start with the facts $\{x' : Q\} \cup \{a \neq b$ for all pairs of abstract individual names appearing in $Q\}$, and the goal $\{x' : V\}$.

The role of the propagation rules is to make explicit (by adding constraints to $\mathcal{Q}$ or $\mathcal{V}$) the part of the knowledge which is implicitly contained in $\mathcal{Q}$, $\mathcal{V}$ and the schema $\mathcal{S}$. They add facts and goals until no rule applies. Intuitively, the view V contains the query Q *if and only if* the final set of facts contains the constraint $x' : V$.

Given a pair of constraint systems $\mathcal{Q}.\mathcal{V}$, more than one rule might be applicable to it. we define the following strategy for the application of rules:

1. apply the decomposition rules as long as possible;
2. apply the schema rules as long as possible;
3. apply the goal rules as long as possible;
4. apply the composition rules.

Before we can formulate the propagation rules we need a technical definition.

Definition 11 (Fork) *Let $\tilde{S}$ be a constraint system, f be a feature name, s and t be abstract individuals (i.e., abstract individual names or variables), z_1 and z_2 be concrete individual names, and x be a variable. $\tilde{S}$ may contain the following constraints, which we call a* fork*:*

- *$(s,t) : f$, and $(s,x) : f$. Since f is interpreted as a partial function, such a fork means that t and x have to be interpreted as the same abstract individual. This fork can be deleted by replacing all occurrences of x in $\tilde{S}$ by t.*
- *$(s,z_1) : f$ and $(s,z_2) : f$. This fork is due to the fact that we do not handle unique name assumption for concrete individuals. This fork can be deleted by replacing all occurrences of z_2 in $\tilde{S}$ by z_1.*

Let R be a role name (resp. f be a feature name). In the following, we write equally sRt or $(s,t) : R$ (resp. sfv or $(s,v) : f$) to denote the fact that t is a R-successor (resp. v is a f-successor) of s.

Decomposition Rules

These rules add constraints to the constraint system $\mathcal{Q}$. They break up the initial fact $x' : Q$ into constraints involving primitive concepts.

D₁) $\mathcal{Q}.\mathcal{V} \quad \longrightarrow_{\sqcap} \quad \{s : C_1, s : C_2\} \cup \mathcal{Q}.\mathcal{V}$
if $s : C_1 \sqcap C_2$ is in $\mathcal{Q}$ and
$s : C_1$ and $s : C_2$ are not both in $\mathcal{Q}$

D₂) $\mathcal{Q}.\mathcal{V} \quad \longrightarrow_{\exists} \quad \{sRy, y : C\} \cup \mathcal{Q}.\mathcal{V}$
if $s : \exists R.C$ is in $\mathcal{Q}$ and
y is a new variable and
there is no t such that t is an
R–successor of s in $\mathcal{Q}$ and $t : C$ is in $\mathcal{Q}$

D₃) $\mathcal{Q}.\mathcal{V} \quad \longrightarrow_{P_r} \quad \{\ldots,(s,y_{i_1}) : f_{i_1},\ldots,(y_{i_{n_i}-1},z_i) : f_{i_{n_i}},\ldots,$
$(z_1,\ldots,z_n) : P_r\} \cup \mathcal{Q}.\mathcal{V}$
if $s : P_r(u_1,\ldots,u_n)$ is in $\mathcal{Q}$, and
$z_1,\ldots,z_n$ are new concrete individual names
and $\ldots,y_{i_1},\ldots,y_{i_{n_i}-1},\ldots,$ are new variables
and the following does not hold
For the feature chains $u_i = f_{i_1}\ldots f_{i_{n_i}}$,
$i \in [1,n]$, there are abstract individual
names $t_{i_1},\ldots,t_{i_{n_i}-1}$ and
a concrete individual name z_i', such that
$\mathcal{Q}$ contains constraints $(s,t_{i_1}) : f_{i_1},\ldots,$
$(t_{i_{n_i}-1},z_i') : f_{i_{n_i}},\ldots,(z_1',\ldots,z_n') : P_r$

We may have created forks by this rule. If this is the case, we delete them as described before.

D$_4$) $\mathcal{Q}.\mathcal{V}$ $\longrightarrow_{\Theta}$ $\{s : C, y : D, \ldots, (s, x_{i_1}) : f_{i_1}, \ldots,$
$(x_{i_{n_i}-1}, z_i) : f_{i_{n_i}}, (y, x'_{i_1}) : f'_{i_1}, \ldots,$
$(x'_{i_{n'_i}-1}, z'_i) : f'_{i_{n'_i}}, (z_i, z'_i) : P_{os_i}, \ldots\} \cup \mathcal{Q}.\mathcal{V}$
if $s : \Theta(C, D, \{\langle u_1, P_{os_1}, u'_1\rangle, \ldots, \langle u_m, P_{os_m}, u'_m\rangle\})$
is in $\mathcal{Q}$, and $z_1, z'_1, \ldots, z_m, z'_m$ are new concrete
individual names and y is a new variable
and $\ldots, x_{i_1}, \ldots, x_{i_{n_i}-1}, x'_{i_1} \ldots, x'_{i_{n'_i}-1}, \ldots,$ are
new variables and the following does not hold
For the feature chains $u_i = f_{i_1} \ldots f_{i_{n_i}}$, and
$u'_i = f'_{i_1} \ldots f'_{i_{n'_i}}$
$i \in [1, m]$ there are abstract individual
names $t_{i_1}, \ldots, t_{i_{n_i}-1}, t'_{i_1}, \ldots, t'_{i_{n'_i}-1}$ and
concrete individual name v_i, v'_i, such that
$\mathcal{Q}$ contains constraints $(s, t_{i_1}) : f_{i_1}, \ldots,$
$(t_{i_{n_i}-1}, v_i) : f_{i_{n_i}}, (s', t'_{i_1}) : f'_{i_1}, \ldots,$
$(t'_{i_{n'_i}-1}, v'_i) : f'_{i_{n'_i}}, (v_i, v'_i) : P_{s_i}$

We may have created forks by this rule. If this is the case, we delete them as described before.

D$_5$) $\mathcal{Q}.\mathcal{V}$ $\longrightarrow_{-}$ $\{tRs\} \cup \mathcal{Q}.\mathcal{V}$
if $sR^{-}t$ is in $\mathcal{Q}$ and tRs is not in $\mathcal{Q}$

D$_6$) $\mathcal{Q}.\mathcal{V}$ $\longrightarrow_{[]}$ $[y/a]\mathcal{Q}.\mathcal{V}[y/a]$
if $y : \{a\}$ is in $\mathcal{Q}$

D$_6$ is a substitution rule. We read $[y/a]$ as "a replaces y". This substitution applies to both constraint systems, i.e., $\mathcal{Q}$ and $\mathcal{V}$.

D$_7$) $\mathcal{Q}.\mathcal{V}$ $\longrightarrow_{P_s^{(\Phi)}}$ $\{\ldots, (s, y_1) : f_1, \ldots, (y_{n-1}, z) : f_n, \ldots,$
$(z) : P_s^{(\Phi)}\} \cup \mathcal{Q}.\mathcal{V}$
if $s : P_s^{(\Phi)}(u)$ is in $\mathcal{Q}$, and
z is a new concrete individual name
and $y_1, \ldots, y_{n-1}$ are new variables
and the following does not hold
For the feature chain $u = f_1 \ldots f_n$,
there are abstract individual
names $t_1, \ldots, t_{n-1}$ and
a concrete individual name z', such that
$\mathcal{Q}$ contains constraints $(s, t_1) : f_1, \ldots,$
$(t_{n-1}, z') : f_n, (z') : P_s^{(\Phi)}$

We may have created a fork by this rule. If this is the case, we delete it as described before.

Schema Rules

These rules add constraints to the constraint system $\mathcal{Q}$. They add information derivable from the schema $\mathcal{S}$ and current facts contained in $\mathcal{Q}$.

S₁) $\mathcal{Q}.\mathcal{V} \longrightarrow_{1.\dot{\preceq}} \{s : D\} \cup \mathcal{Q}.\mathcal{V}$
if $s : A$ is in $\mathcal{Q}$, $A \dot{\preceq} D$ is in $\mathcal{S}$ and
$s : D$ is not in $\mathcal{Q}$

S₂) $\mathcal{Q}.\mathcal{V} \longrightarrow_{\forall} \{v : A'\} \cup \mathcal{Q}.\mathcal{V}$
if $s : A, s\,P\,v$ are in $\mathcal{Q}$, $A \dot{\preceq} \forall P.A'$ is in $\mathcal{S}$
(resp. $s : A, sfv$ are in $\mathcal{Q}$ and
$A \dot{\preceq} \forall f.A'$ is in $\mathcal{S}$) and $v : A'$ is not in $\mathcal{Q}$

S₃) $\mathcal{Q}.\mathcal{V} \longrightarrow_{2.\dot{\preceq}} \{s : A, v : A'\} \cup \mathcal{Q}.\mathcal{V}$
if sPv is in $\mathcal{Q}$ and $P \dot{\preceq} A \times A'$ is in $\mathcal{S}$
(respectively sfv is in $\mathcal{Q}$ and $f \dot{\preceq} A \times A'$ is in $\mathcal{S}$)
and $s : A, v : A'$ are not both in $\mathcal{Q}$

S₄) $\mathcal{Q}.\mathcal{V} \longrightarrow_{\exists} \{s\,P\,y, y : C\} \cup \mathcal{Q}.\mathcal{V}$
if $s : A$ is in $\mathcal{Q}$, $A \dot{\preceq} \exists P.\top$ is in $\mathcal{S}$, $s : \exists P.C$ is in $\mathcal{V}$
and there is no v' such that $s\,P\,v', v' : C$
are in $\mathcal{Q}$ and y is a new variable

S₅) $\mathcal{Q}.\mathcal{V} \longrightarrow_{1.\exists} \{s\,f\,v, v : C\} \cup \mathcal{Q}.\mathcal{V}$
if $s : A$ is in $\mathcal{Q}$, $A \dot{\preceq} \exists f.\top$ is in $\mathcal{S}$, $s : \exists f.C$ is in
$\mathcal{V}$ and there is no v' such that
$s\,fv', v' : C$ are in $\mathcal{Q}$ and
v is a new variable if C is a concept and
v is a new concrete individual name if C
is a concrete domain name

We may have created a fork by this rule. If this is the case, we delete it as described before.

S₆) $\mathcal{Q}.\mathcal{V} \longrightarrow_{2.\exists} \{s\,f\,v, v : C, v : A'\} \cup \mathcal{Q}.\mathcal{V}$
if $s : A$ is in $\mathcal{Q}$, $A \dot{\preceq} \exists f.A'$ is in $\mathcal{S}$, $s : \exists f.C$
is in $\mathcal{V}$ and there is no v' such that $s\,fv', v' : C$,
$v' : A'$ are in $\mathcal{Q}$ and
v is a new variable if C is a concept and
v is a new concrete individual name if C
is a concrete domain name

We may have created a fork by this rule. If this is the case, we delete it as described before.

Goal Rules

These rules add constraints to the constraint system $\mathcal{V}$. They guide the evaluation of V by deriving subgoals from the original goal $x' : V$.

G₁) $\mathcal{Q}.\mathcal{V} \longrightarrow_{\sqcap} \mathcal{Q}.\mathcal{V} \cup \{s : C_1, s : C_2\}$
if $s : C_1 \sqcap C_2$ is in $\mathcal{V}$ and
$s : C_1, s : C_2$ are not in $\mathcal{Q} \cup \mathcal{V}$

G₂) $\mathcal{Q}.\mathcal{V} \longrightarrow_{\exists} \mathcal{Q}.\mathcal{V} \cup \{t : C\}$
if $s : \exists R.C$ is in $\mathcal{V}$ and sRt is in $\mathcal{Q}$ and
$t : C$ is not in $\mathcal{Q} \cup \mathcal{V}$

G₃) $\mathcal{Q}.\mathcal{V} \longrightarrow_{1.\Theta} \mathcal{Q}.\mathcal{V} \cup \{s : C\}$
if $s : \Theta(C, D, \{\langle u_1, P_{os_1}, u_1' \rangle, \ldots,$
$\langle u_m, P_{os_m}, u_m' \rangle\})$
is in $\mathcal{V}$ and $s : C$ is not in $\mathcal{Q} \cup \mathcal{V}$

G$_4$) $\mathcal{Q}.\mathcal{V} \quad \longrightarrow_{2.\Theta} \quad \mathcal{Q}.\mathcal{V} \cup \{t : D\}$
if $s : \Theta(C, D, \{\langle u_1, P_{os_1}, u_1^{'}\rangle, \ldots, \langle u_m, P_{os_m}, u_m^{'}\rangle\})$ is in $\mathcal{V}$ and
$\ldots, (s, x_{i_1}) : f_{i_1}, \ldots, (x_{i_{n_i}-1}, z_i) : f_{i_{n_i}}$,
$(y, x_{i_1}^{'}) : f_{i_1}^{'}, \ldots, (x_{i_{n_i^{'}}-1}^{'}, z_i^{'}) : f_{i_{n_i^{'}}}^{'}$,
$(z_i, z_i^{'}) : P_{os_i}, \ldots$ for $i \in [1, n]$ are in $\mathcal{Q}$ and $t : D$ is not in $\mathcal{Q} \cup \mathcal{V}$

Composition Rules

These rules add constraints to the constraint system $\mathcal{Q}$. They compose[9] complex facts from simpler ones directed by the goals.

C$_1$) $\mathcal{Q}.\mathcal{V} \quad \longrightarrow_{\sqcap} \quad \{s : C_1 \sqcap C_2\} \cup \mathcal{Q}.\mathcal{V}$
if $s : C_1, s : C_2$ are in $\mathcal{Q}$, and
$s : C_1 \sqcap C_2$ is in $\mathcal{V}$, but not in $\mathcal{Q}$

Let $\mathcal{D}$ be a concrete domain, $u_1, \ldots, u_n$ be feature chains, and P_r, P_r' be predicate names in $\mathsf{pred}(\mathcal{D})$. $P_r'(u_1, \ldots, u_n)$ *entails* $P_r(u_1, \ldots, u_n)$ iff $\forall e_1, \ldots, e_n \in \mathsf{dom}(\mathcal{D}), (e_1, \ldots, e_n) \in {P_r'}^{\mathcal{D}} \Rightarrow (e_1, \ldots, e_n) \in {P_r}^{\mathcal{D}}$. We are able to decide this because we have supposed that the *implication* between finite conjunctions over $\mathsf{pred}(\mathcal{D})$ is *decidable.*

C$_2$) $\mathcal{Q}.\mathcal{V} \quad \longrightarrow_{P_r} \quad \{s : P_r(u_1, \ldots, u_n)\} \cup \mathcal{Q}.\mathcal{V}$
if $s : P_r'(u_1, \ldots, u_n)$ is in $\mathcal{Q}$ and
$s : P_r(u_1, \ldots, u_n)$ is in $\mathcal{V}$ but not in $\mathcal{Q}$, and
$P_r'(u_1, \ldots, u_n)$ entails $P_r(u_1, \ldots, u_n)$

C$_3$) $\mathcal{Q}.\mathcal{V} \quad \longrightarrow_{\Theta} \quad \{s : \Theta(C, D, \{\langle u_1, P_{os_1}, u_1^{'}\rangle, \ldots, \langle u_m, P_{os_m}, u_m^{'}\rangle)\} \cup \mathcal{Q}.\mathcal{V}$
if $s : C$ and $t : D$ are in $\mathcal{Q}$ and
$(s, x_{i_1}) : f_{i_1}, \ldots, (x_{i_{n_i}-1}, z_i) : f_{i_{n_i}}$,
$(t, x_{i_1}^{'}) : f_{i_1}^{'}, \ldots, (x_{i_{n_i^{'}}-1}^{'}, z_i^{'}) : f_{i_{n_i^{'}}}^{'}$,
$(z_i, z_i^{'}) : P_{os_i}$ are in $\mathcal{Q}$ for each $i \in [1, m]$
where $u_i = f_{i_1} \ldots f_{i_{n_i}}$ and $u_i^{'} = f_{i_1}^{'} \ldots f_{i_{n_i^{'}}}^{'}$ and
$s : \Theta(C, D, \{\langle u_1, P_{os_1}, u_1^{'}\rangle, \ldots, \langle u_m, P_{os_m}, u_m^{'}\rangle)$ is in $\mathcal{V}$ but not in $\mathcal{Q}$

C$_4$) $\mathcal{Q}.\mathcal{V} \quad \longrightarrow_{\top} \quad \{s : \top\} \cup \mathcal{Q}.\mathcal{V}$
if $s : \top$ is in $\mathcal{V}$ but not in $\mathcal{Q}$

C$_5$) $\mathcal{Q}.\mathcal{V} \quad \longrightarrow_{\exists} \quad \{s : \exists R.C\} \cup \mathcal{Q}.\mathcal{V}$
if $s : \exists R.C$ is in $\mathcal{V}$ but not in $\mathcal{Q}$ and
$sRt, t : C$ are in $\mathcal{Q}$

All rules are deterministic. Moreover, rules **D$_2$**, **D$_3$**, **D$_4$**, **D$_7$**, **S$_4$**, **S$_5$**, **S$_6$** are generating ones, since they generate variables or concrete individual names.

A constraint system $\tilde{S}$ is *complete* if no propagation rule applies to it. A constraint system contains a clash if it displays one of the following situations:

- It contains the constraints

[9]This can be seen as a bottom up evaluation of V over $\mathcal{Q}$.

$(z_1^{(1)}, \ldots, z_{n_1}^{(1)}) : P_{r_1}, \ldots, (z_1^{(k)}, \ldots, z_{n_k}^{(k)}) : P_{r_k}$ and

$$\bigwedge_{i=1}^{k} P_{r_i}^{\mathcal{D}}(\bar{z}^{(i)}) \text{ is not satisfiable in a concrete domain } \mathcal{D}.$$

- It contains the constraints $s : \{t\}, s \neq t$.
- It contains the constraints sft, sfz, where t is an abstract individual and z is a concrete individual name.
- It contains the constraint $s : \mathsf{dom}(\mathcal{D})$, where s is an abstract individual.
- It contains the constraint $z : A$ where z is a concrete individual name and A a concept name.

Therefore, any constraint system containing a clash is unsatisfiable.

In the following, x' is a variable, $\mathcal{F}_Q.\mathcal{G}_\mathcal{V}$ is the *completion* of $\{x' : Q\}.\{x' : V\}$, and o is an abstract individual name such that $o : V$ is in $\mathcal{G}_\mathcal{V}$.

Let $\mathcal{F}_Q.\mathcal{G}_\mathcal{V}$ be a complete pair derivable from an initial pair $\{x' : Q\}.\{x' : V\}$. By construction, $\mathcal{G}_\mathcal{V}$ contains exactly one constraint of the form $s : V$. In addition, as goal rules are not generating ones and by examining all other rules, we observe that if $s : V \in \mathcal{G}_\mathcal{V}$ then $s : Q \in \mathcal{F}_Q$.

Proposition 2 (Invariance) *Suppose $\mathcal{F}.\mathcal{G}$ has been derived from $\{x' : Q\}.\{x' : V\}$, and $\mathcal{F}'.\mathcal{G}'$ is obtained from $\mathcal{F}.\mathcal{G}$ by applying a rule. Then $\mathcal{F}$ is satisfiable if and only if $\mathcal{F}'$ is satisfiable.*

Proof See Appendix.

Corollary 1 *Every $\mathcal{S}$-model $\mathcal{J}$ of $x' : Q$ can be turned into an $\mathcal{S}$-model $\mathcal{J}'$ of $\mathcal{F}_Q$ by modifying the interpretation of variables and concrete individual names. Moreover, $\mathcal{J}'$ can be chosen such that $o^{\mathcal{J}'} = x'^{\mathcal{J}}$.*

Proof It follows by induction from the preceding Proposition. ■

Corollary 2 *Let $\mathcal{F}_Q$ be the complete constraint system derived from $\{x' : Q\}$, and let o be an abstract individual name. The following holds:*

$$x' : Q \models_\mathcal{S} x' : V \Leftrightarrow \mathcal{F}_Q \models_\mathcal{S} o : V$$

Proof See Appendix.

Let $\tilde{S}$ be a clash-free constraint system. We define the canonical interpretation $\mathcal{J}_{\tilde{S}}$ as follows:

- Because the clash rule related to concrete domains is not applicable, there is an assignment α that *satisfies* the conjunction of all occurring constraints of the form $P_r(z_1, \ldots, z_n)$. The interpretation $\mathcal{J}_{\tilde{S}}$ interprets a concrete individual name z as $\alpha(z)$.
- the domain $\Delta^{\mathcal{J}_{\tilde{S}}}$ consists of all abstract individuals occurring in $\tilde{S}$.
- Let A be a primitive concept name. Then we set $s \in A^{\mathcal{J}}$ iff $s : A$ occurs in $\tilde{S}$.

- Let R be a role or a feature name. Then we set $(s, v) \in R^{\mathcal{J}_{\tilde{S}}}$ iff $(a, v) : R$ occurs in $\tilde{S}$. This is well defined even if R is a feature, because there is no clash related to the features. Here v is an abstract individual or a concrete individual name.

Proposition 3 *Let $\mathcal{F}_Q.\mathcal{G}_\mathcal{V}$ be a complete pair that has been derived from $\{x' : Q\}.\{x' : V\}$. If $\mathcal{F}_Q$ is clash-free, then the canonical interpretation $\mathcal{J}_{\mathcal{F}_Q}$ is an $\mathcal{S}$-model of $\mathcal{F}_Q$.*

Proof See Appendix.

Proposition 4 *Let $\mathcal{J}_{\mathcal{F}_Q}$ be the canonical interpretation of $\mathcal{F}_Q$ and $s : C$ be a constraint in $\mathcal{G}_\mathcal{V}$. If $\mathcal{F}_Q$ is clash-free, then*

$$\mathcal{J}_{\mathcal{F}_Q} \text{ satisfies } s : C \Longrightarrow s : C \in \mathcal{F}_Q$$

Proof See Appendix.

Definition 12 (Size of a feature chain) *Let $u = f_1 \dots f_r$ be a feature chain. Then $|u| = r$.*

Definition 13 (Size of a concept) *For a concept C, the size $|C|$ is inductively defined as:*

- $|P_r(u_1, \dots, u_n)| = \sum_{i=1}^{n} |u_i|$ *for all n-ary predicates of the concrete domains and features $u_1, \dots, u_n$.*
- $|P_s^{(\Phi)}(u)| = |u|$.
- $|\Theta(C, D, \{\langle u_1, P_{os_1}, u_1^{'}\rangle, \dots, \langle u_m, P_{os_m}, u_m^{'}\rangle\})| = |C| + |D| + \Sigma_{i=1}^{m}|u_i| + \Sigma_{i=1}^{m}|u_i^{'}|$.
- $|A| = 1$,
- $|\top| = 1$,
- $|\exists R.C| = 1 + |C|$.
- $|\exists R.C| = 1 + |C|$,
- $|\{a\}| = 1$,
- $|C \sqcap D| = |C| + |D|$.

The size of a schema $\mathcal{S}$ is given by the number of axioms in $\mathcal{S}$.

Proposition 5 (Termination) *Let Q and V be a query and a view respectively. Then there is no infinite chain of completion steps issuing from $\{x' : Q\}$ and $\{x' : V\}$.*

Proof See Appendix.

Theorem 1 (Soundness and Completeness)

$$Q \dot{\preceq}_\mathcal{S} V \text{ iff } o : V \in \mathcal{F}_Q \text{ or } \mathcal{F}_Q \text{ contains a clash}$$

Proof See Appendix.

Now we turn to the complexity of deciding $\mathcal{S}$-containment.

Proposition 6 (Number of individuals) *The number of individuals occurring in $\mathcal{F}_Q.\mathcal{G}_\mathcal{V}$ is bounded by $|Q|.|V|$.*

Proof See Appendix.

Theorem 2 *$\mathcal{S}$-containment of a query Q in a view V can be decided in time polynomial to the size of Q, V, and $\mathcal{S}$.*

Proof See Appendix.

5 Comparisons with other works

Our work relates to several aspects of modeling and retrieval of visual information. We shortly discuss the relationship to shape representation for image retrieval and knowledge representation approaches to visual information retrieval.

Shape representation for image retrieval. In [8], the authors have proposed to capture objects shapes in image databases by means of two complementary methods namely, Freeman code and Fourier descriptors. The first one applies to closed shapes by approximation of the continuous contour with a sequence of numbers, each number corresponding to a segment direction. This approach is not suited for complex shapes. The second method rests on the use of complex coefficients called Fourier descriptors. These coefficients represent the shape of an object in the frequency domain where the lower frequencies symbolize its general contour, and where the higher frequencies represent the details of its contour. Clearly, these two methods are not suitable for large databases. [22] has proposed a two-block data model for images: The image block and the salient object bloc. The image block is made up of two layers: the image layer and the image representation layer. The geometric primitives used to specify objects shapes are the conventional ones, namely, point, segment, polyline, ellipse, circle, polygon, triangle, and square. It is clear that the constraint-based approach we presented for representing objects shapes generalizes [22] since all the geometric primitives can be naturally defined by means of our constraints. In [5], a constraint-based approach[10] is used for shape management in multimedia databases. The advantage of this approach is that approximation-based query processing, combined with data-driven approaches can be used to retrieve shapes based on similarity. However, reasoning about queries (containment and emptyness) is not considered.

Knowledge representation approaches to visual information retrieval. Meghini *et al.* [12] have investigated the use of a description logic as a conceptual tool for modeling and querying image data. Their language is a fragment of $\mathcal{ALC}$[23] extended to accommodate fuzzy aspects. The visual part in a query is captured through a mechanism of procedural attachment which is a kind of logical interface between the conceptual part and the visual part of an image. The problem with this language is that subsumption between concepts is $PSPACE$-complete. Hsu *et al.* [13] proposed a knowledge-based approach for retrieving images by content. The knowledge-based query processing is based on a query relaxation technique which exploits a Type Abstraction Hierarchy of image features. Goble *et al.* [17] proposed a description logic, called, GRAIL, for describing the image and video semantic content. A set of dedicated constructors are used to capture the structural part of these media objects. The aim is to support the coherent and incremental development of a coarse index on the semantic annotations of media documents. In these proposals, the underlying query languages support only queries based on the structure of the documents (i.e., conceptual queries). None of them supports visual queries. Additionally, [12] and [17] do not take

[10]In the sense of constraint relational calculus [19].

into account predicate restrictions over concrete domains, which are extremely useful when querying multimedia repositories, and they did not address the questions of decidability and complexity of reasoning services in their languages.

6 Conclusion

The bulk of work in image processing research has been on developing algorithms that operate at the pixel level and are able to recognize visual objects. In our work, we have been investigating the next stage of image processing. In other words, we are concerned with the question of what sort of representation and processing would we like to have happen once the low-level detectors have finished their work. We feel that the next step involves representing and reasoning with the aid of formal models. By increasing the level of abstraction and allowing queries at that level, it becomes easier to express queries for finding appropriate data for viewing out of a large repository. An interesting problem is the investigation and adaptation of optimization techniques for constraint query languages (see, for example, [24]).

References

[1] Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases.* Addison-Wesley, 1995.

[2] Franz Baader, Martin Bucheit, Manfred Jeusfeld, and Werner Nutt. Reasoning about Structured Objects: Knowledge Representation meets Databases. In F. Baader, M. Bucheit, M. Jeusfeld, and W. Nutt, editors, *Proceedings of the* 1^{st} *Workshop of KRDB'94: Reasoning about Structured Objects: Knowledge Representation meets Databases, Stuhlsatzenhausweg, Germany*, D-94-11 in DFKI Documents, URL: http://SunSite.Informatik.RWTH-Aachen.DE/Publications/CEUR-WS/Vol-1/, September 1994. CEUR. 2p.

[3] Franz Baader and Philipp Hanschke. A Scheme for Integrating Concrete Domains into Concept Languages. In *Proceedings of the 12th International Joint Conference on Artificial Intelligent (IJCAI'91). Sydney, Australia*, pages 452–457, 1991.

[4] Riccardo Benedetti and Jean-Jacques Risler. *Real Algebraic and Semi-Algebraic Sets.* Hermann, editeurs des sciences et des arts, 293 rue Lecourbe, 75015 Paris, 1990. 340 pages.

[5] E. Bertino and B. Catania. A Constraint-Based Approach to Shape Management in Multimedia Databases. *Multimedia Systems*, 6(1):2–16, January 1998.

[6] Alberto Del Bimbo. *Visual Information Retrieval.* Morgan Kaufmann, 1999.

[7] Alexander Borgida. Description Logics in Data Management. *IEEE Transactions on Knowledge and Data Engineering (IEEE TKDE)*, 7(7):671–682, 1995.

[8] M. Bouet, A. Khenchaf, and H. Briand. Shape Representation for Image Retrieval. In *Proceedings of the Seventh ACM International Conference on Multimedia, Orlando, FL, USA*, pages 1–4, October 30 - November 5 1999.

[9] Ronard J. Brachman and James G. Schmolze. An Overview of the KL-ONE Knowledge Representation System. *Cognitive Science*, 9(2):171–216, 1985.

[10] P. Bresciani. Some Research Trends in KR&DB (position paper). In F. Baader, M. Bucheit, M. Jeusfeld, and W. Nutt, editors, *Proceedings of the* 3^{td} *Workshop of KRDB'96: Reasoning about Structured Objects: Knowledge Representation meets Databases, Budapest, Hungary*, pages 1–3, URL: http://SunSite.Informatik.RWTH-Aachen.DE/Publications/CEUR-WS/Vol-4/, August 1996. CEUR.

[11] Martin Buchheit, Manfred A. Jeusfeld, Werner Nutt, and Martin Staudt. Subsumption Between Queries to Object-Oriented Databases. In *Proceedings of the 4th International Conference on Extending Database Technology (EDBT'94), Cambridge, UK*, March 1994. (Also in Information Systems 19(1), pp. 33-54, 1994).

[12] Fabrizio Sebastiani Carlo Meghini and Umberto Straccia. The Terminological Image Retrieval Model. In Alberto Del Bimbo, editor, *Proceedings of the 9th International Conference On Image Analysis And Processing (ICIAP'97), Florence, Italy*, pages 156–163, 1997.

[13] Wesley W. Chu Chih-Cheng Hsu and Ricky K. Taira. A Knowledge-Based Approach for Retrieving Images by Content. *IEEE Transactions on Knowledge and Data Engineering*, 8(4):522–532, 1996.

[14] George E. Collins. Quantifier Elemination for Real Closed Fields by Cylindrical Algebraic Decomposition. In *Proceedings of the 2nd Conference on Automata Theory & Formal Languages, Kaiserslautern, Germany*, pages 134–183, 1975. Volume 33 of LNCS.

[15] Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, and Werner Nutt. The Complexity of Concept Languages. Technical Report RR-95-07, Deutsches Forschunggszentrum für Künstliche Intelligenz (DFKI), Kaiserslautern, Germany, June 1995.

[16] Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, and Andrea Schaerf. Reasoning in Description Logics. In *Foundation of Knowledge Representation*. Cambrige University Press, 1995.

[17] C. A. Goble, C. Haul, and S. Bechhofer. Describing and Classifying Multimedia Using the Description Logic GRAIL. In Ishwar K. Sethi and Ramesh C. Jain, editors, *Storage and retrieval for image and video database IV (SPIE'96), San Jose, California*, pages 132–143, February 1996.

[18] Stéfane Grumbach, Jianwen Su, and Christophe Tollu. Linear Constraint Query Languages Expressive Power and Complexity. In Daniel Leivant, editor, *Proceedings of the International Workshop on Logic and Computational Complexity (LCC'94), Indianapolis, IN, USA*, pages 426–446, October.

[19] Paris Kanellakis, Gabriel Kuper, and P. Revesz. Constraint Query Languages. *Journal of Computer and System Sciences (JCSS)*, 51(1):26–52, August 1995.

[20] Carlo Meghini. Towards a logical reconstruction of image retrieval. In Ishwar K. Sethi and Ramesh C. Jain, editors, *Storage and retrieval for image and video database IV (SPIE'96), San Jose, California*, pages 108–119, February 1996.

[21] B. Nebel. Reasoning and Revision in Hybrid Representation Systems. volume 422 of *Lecture Notes in Computer Science*, page 300. Springer-Verlag, New York, 1990.

[22] V. Oria, M.T. Ozsu, L. I. Cheng, P. J. Iglinski, and Y. Leontiev. Modeling and Querying Shapes in Image Database System. In *Proceedings of the Fifth International Workshop on Multimedia Information Systems (MIS'99), Indian Wells, Palm Springs Desert, CA, USA*, October 21 - 23 1999.

[23] Manfred Schmidt-Schauß and Gert Smolka. Attributive Concept Descriptions with Complements. *Artificial Intelligence*, 48(1):1–26, 1991.

[24] Divesh Srivastava. Subsumption and Indexing in Constraint Query Languages with Linear Arithmetic Constraints. *Annals of Mathematics and Artificial Intelligence*, 8(3-4):315–343, 1993.

[25] Alfred Tarski. *A Decision Method for Elementary Algebra and Geometry*. University of California Press. Berkeley, 1951.

[26] Jeffrey D. Ullman. *Principles of Database and Knowledge-Base Systems*, volume I, II. Computer Science Press, Rockville MD, 1989.

[27] Luc Vandeurzen, Marc Gyssens, and Dirk Van Gucht. On Query Languages for Linear Queries Definable with Polynomial Constraints. In *Proceedings of the Second International Conference on Principles and Practice of Constraint Programming (CP'96), Cambridge, Massachusetts, USA*, pages 468–481. Springer Verlag, August 1996. LNCS 1118.

Appendix

In this Appendix we give the proofs of some of the results stated in the previous sections.

Proof (Proposition 1)
" $\Rightarrow$ "
If $Q\dot{\preceq}_{\mathcal{S}}V$ then $Q^{\mathcal{J}} \subseteq V^{\mathcal{J}}$ for all model $\mathcal{J}$ of $\mathcal{S}$. This means that if x' denotes an individual from the interpretation domain of $\mathcal{J}$ such t hat $x'^{\mathcal{J}} \in Q^{\mathcal{J}}$, then $x'^{\mathcal{J}} \in V^{\mathcal{J}}$. It follows that $x' : Q \models_{\mathcal{S}} x' : V$.

" $\Leftarrow$ "
Suppose that $x'^{\mathcal{J}} \in Q^{\mathcal{J}}$ implies $x'^{\mathcal{J}} \in V^{\mathcal{J}}$ for all $\mathcal{J}$ model of $\mathcal{S}$. It follows that $Q^{\mathcal{J}} \subseteq_{\mathcal{S}} V^{\mathcal{J}}$. Hence, $Q\dot{\preceq}_{\mathcal{S}}V$. ■

Proof (Proposition 2)

The proof is by case analysis.

" $\Rightarrow$ "

Let $\mathcal{J}$ be an $\mathcal{S}$-model of $\mathcal{F}$. Then $\mathcal{J}$ can be turned into an $\mathcal{S}$-model of $\mathcal{F}'$ by modifying the interpretation of new variables and new concrete individual names. We have to consider all the rules that alter the set of facts (i.e., decomposition, schema, and composition rules).

D₁) $\mathcal{J}$ satisfies $s : C_1 \sqcap C_2$. That is $s^{\mathcal{J}} \in (C_1 \sqcap C_2)^{\mathcal{J}}$. This means that $s^{\mathcal{J}} \in (C_1^{\mathcal{J}} \cap C_2^{\mathcal{J}})$, and then $s^{\mathcal{J}} \in C_1^{\mathcal{J}}$ and $s^{\mathcal{J}} \in C_2^{\mathcal{J}}$. Hence $\mathcal{J}$ satisfies $s : C_1$ and $s : C_2$. It follows that $\mathcal{J}$ is an $\mathcal{S}$-model for $\mathcal{F}'$.

D₂) $\mathcal{J}$ satisfies $s : \exists R.C$. There exists an abstract individual t such that t is an R-successor of s and $t^{\mathcal{J}} \in C^{\mathcal{J}}$. It follows that $\mathcal{J}$ is an $\mathcal{S}$-model for $\mathcal{F}'$.

D₃) $\mathcal{J}$ satisfies $s : P_r(u_1, \ldots, u_n)$. That is, if $z_1, \ldots, z_n$ are concrete individual names such that $u_1^{\mathcal{J}}(s^{\mathcal{J}}) = z_1^{\mathcal{J}}, \ldots, u_n^{\mathcal{J}}(s^{\mathcal{J}}) = z_n^{\mathcal{J}}$ we have $(z_1^{\mathcal{J}}, \ldots, z_n^{\mathcal{J}}) \in {P_r}^{\mathcal{D}}$. It follows that $\mathcal{J}$ is an $\mathcal{S}$-model for $\mathcal{F}'$.

D₄) $\mathcal{J}$ satisfies
$s : \Theta(C, D, \{\langle u_1, P_{os_1}, u_1^{'}\rangle, \ldots, \langle u_m, P_{os_m}, u_m^{'}\rangle\})$. Then $s^{\mathcal{J}} \in C^{\mathcal{J}}$ and there exists an abstract individual t such that $t^{\mathcal{J}} \in D^{\mathcal{J}}$ and $(u_i^{\mathcal{J}}(s^{\mathcal{J}}), u_i^{'\mathcal{J}}(t^{\mathcal{J}})) \in P_{os_i}^{\mathcal{J}}$ $\forall i \in [1, m]$ where u_i and $u_i^{'}$ are feature chains. It follows that $\mathcal{J}$ is an $\mathcal{S}$-model for $\mathcal{F}'$.

D₅) $\mathcal{J}$ satisfies $sR^{-}t$. Then by definition it satisfies tRs. Hence $\mathcal{J}$ is an $\mathcal{S}$-model for $\mathcal{F}'$.

D₆) Obvious.

D₇) $\mathcal{J}$ satisfies $s : P_s^{(\Phi)}(u)$. This means that $u^{\mathcal{J}}(s^{\mathcal{J}})$ satisfies $P_s^{(\Phi)}$ in $\top_{\mathcal{SAS}}$, where u is a feature chain. In other words, for $u = f_1 \ldots f_n$, there exist abstract individuals $t_1, \ldots, t_{n-1}$, and a concrete individual name z, such that $f_1^{\mathcal{J}}(s^{\mathcal{J}}) = t_1^{\mathcal{J}}, \ldots, f_n^{\mathcal{J}}(t_{n-1}^{\mathcal{J}}) = z^{\mathcal{J}}$ and $z^{\mathcal{J}}$ satisfies $P_s^{(\Phi)}$ in $\top_{\mathcal{SAS}}$. It follows that $\mathcal{J}$ is an $\mathcal{S}$-model for $\mathcal{F}'$.

C₁) It follows from the one of **D₁**.

C₂) $\mathcal{J}$ satisfies $s : P_r'(u_1, \ldots, u_n)$. That is, there exist concrete individual names $z_1, \ldots, z_n$ such that $u_1^{\mathcal{J}}(s^{\mathcal{J}}) = z_1^{\mathcal{J}}$, ..., $u_n^{\mathcal{J}}(s^{\mathcal{J}}) = z_n^{\mathcal{J}}$ and $(z_1^{\mathcal{J}}, \ldots, z_n^{\mathcal{J}}) \in {P_r'}^{\mathcal{D}}$. It follows that $(z_1^{\mathcal{J}}, \ldots, z_n^{\mathcal{J}}) \in {P_r}^{\mathcal{D}}$ for all P_r such that $P_r'(u_1, \ldots, u_n)$ *entails* $P_r(u_1, \ldots, u_n)$. Hence $\mathcal{J}$ is an $\mathcal{S}$-model for $\mathcal{F}'$.

C₃) $\mathcal{J}$ satisfies $s : C, t : D, (u_1(s), u_1^{'}(t)) : P_{os_1}$, ..., $(u_m(s), u_m^{'}(t)) : P_{os_m}$, where u_i and $u_i^{'}$ are feature chains $\forall i \in [1, m]$. Then by definition it satisfies $s : \Theta(C, D, \{\langle u_1, P_{os_1}, u_1^{'}\rangle, \ldots, \langle u_m, P_{os_1}, u_m^{'}\rangle\})$. It follows that $\mathcal{J}$ is an $\mathcal{S}$-model for $\mathcal{F}'$.

C₄) Goal rules are not generating ones. Hence if $s : \top$ is in $\mathcal{V}$ then s appears in $\mathcal{Q}$, and then in $\mathcal{F}'$. As all abstract individuals are instances of $\top$ it follows that $\mathcal{J}$ is an $\mathcal{S}$-model for $\mathcal{F}'$.

C₅) $\mathcal{J}$ satisfies sRt and $t : C$. By definition it satisfies $s : \exists R.C$. It follows that $\mathcal{J}$ is an $\mathcal{S}$-model for $\mathcal{F}'$.

S$_1$) $\mathcal{J}$ satisfies $s : A$. As $\mathcal{J}$ is an $\mathcal{S}$-model of $\mathcal{F}$, it satisfies all the axioms in $\mathcal{S}$. Hence $\mathcal{J}$ satisfies $A \dot{\preceq} A'$. That is, $s^{\mathcal{J}} \in A'^{\mathcal{J}}$. It follows that $\mathcal{J}$ is an $\mathcal{S}$-model for $\mathcal{F}'$.

The other cases require similar reasoning and are therefore omitted.

" $\Leftarrow$ "

The propagation rules add (and never remove) constraints to (from) a constraint system. If $\mathcal{F}'$ is obtained from $\mathcal{F}$ by applying a rule, then $\mathcal{F}'$ is a superset of $\mathcal{F}$. It follows that if $\mathcal{J}$ is an $\mathcal{S}$-model for $\mathcal{F}'$, it is also an $\mathcal{S}$-model for $\mathcal{F}$. ■

Proof (Corollary 2)

First, note that by examining all the rules, we remark that if $s : V$ is in $\mathcal{G}_{\mathcal{V}}$ then $s : Q$ is in $\mathcal{F}_{Q}$. In addition, there is only one constraint of the form $s : V$ in $\mathcal{G}_{\mathcal{V}}$.

" $\Rightarrow$ "
Let $\mathcal{J}$ be an $\mathcal{S}$-model of $\mathcal{F}_Q$. Since $\mathcal{F}_Q$ is a complete system, it contains $o : Q$. Hence, $\mathcal{J}$ is an $\mathcal{S}$-model of $o : Q$. Let us consider $\mathcal{J}'$ such that $x'^{\mathcal{J}'} = o^{\mathcal{J}}$ and $\mathcal{J}'$ coincides with $\mathcal{J}$ otherwise. $\mathcal{J}'$ is an $\mathcal{S}$-model of $x' : Q$ and then of $x' : V$. If $\mathcal{J}'$ is an $\mathcal{S}$-model of $x' : V$ then it is also an $\mathcal{S}$-model of $o : V$. As $\mathcal{J}'$ is chosen such that $x'^{\mathcal{J}'} = o^{\mathcal{J}}$ and $\mathcal{J}$ and $\mathcal{J}'$ coincide on all other symbols, we conclude that $\mathcal{J}$ is an $\mathcal{S}$-model of $o : V$.

" $\Leftarrow$ "
Let $\mathcal{J}$ be an $\mathcal{S}$-model of $x' : Q$. As $\mathcal{F}_Q$ is a complete system of $x' : Q$, we can build an $\mathcal{S}$-model $\mathcal{J}'$ for $\mathcal{F}_Q$, from $\mathcal{J}$, with $o^{\mathcal{J}'} = x'^{\mathcal{J}}$ and by modifying the interpretation of the new generated variables and concrete individual names. By hypothesis $\mathcal{J}'$ is also an $\mathcal{S}$-model of $o : V$. We have $V^{\mathcal{J}} = V^{\mathcal{J}'}$ and $o^{\mathcal{J}'} = x'^{\mathcal{J}}$. Hence we can conclude that $\mathcal{J}$ is also an $\mathcal{S}$-model of $x' : V$. ■

Proof (Proposition 3)

We have to verify that $\mathcal{J}_{\mathcal{F}_Q}$ satisfies every axiom in $\mathcal{S}$ and every constraint in $\mathcal{F}_Q$.
First, consider the schema axioms. Suppose that $\mathcal{S}$ contains $A \dot{\preceq} \forall P.A'$. Let $s \in A^{\mathcal{J}_{\mathcal{F}_Q}}$ and $(s, v) \in P^{\mathcal{J}_{\mathcal{F}_Q}}$. Then there is a constraint $v : A'$ in $\mathcal{F}_Q$ since otherwise rule **S$_2$** would be applicable. Thus the axiom is satisfied. Suppose that $\mathcal{S}$ contains $A \dot{\preceq} \exists f.A'$. Let $s \in A^{\mathcal{J}_{\mathcal{F}_Q}}$. Then there are constraints $sfv, v : A'$ in $\mathcal{F}_Q$ since otherwise rule **S$_4$** would be applicable. Thus the axiom is satisfied. We use a similar reasoning for the other forms of axioms.
Next we consider the different constraints in $\mathcal{F}_Q$. By definition of $\mathcal{J}_{\mathcal{F}_Q}$, every constraint $s : A$, $s : \top$, $(s, t) : R$ is satisfied. To prove that more complex constraints are satisfied, we proceed by induction. Suppose $\mathcal{F}_Q$ contains $s : P_r(u_1, \ldots, u_n)$. Then because of the rule **D$_3$** it contains as well $s\, u_1\, z_1, \ldots, s\, u_n\, z_n, (z_1, \ldots, z_n) : P_r$ which are satisfied by inductive hypothesis. Hence, $\mathcal{J}_{\mathcal{F}_Q}$ satisfies also $s : P_r(u_1, \ldots, u_n)$.
Suppose $\mathcal{F}_Q$ contains
$s : \Theta(C, D, \{\langle u_1, P_{os_1}, u_1^{'}\rangle, \ldots, \langle u_m, P_{os_m}, u_m^{'}\rangle\})$. Then because of the rule **D$_4$** it contains as well $s : C$, $y : D$, …, $(s, x_{i_1}) : f_{i_1}$, …, $(x_{i_{n_i}-1}, z_i) : f_{i_{n_i}}$, $(y, x_{i_1}^{'}) : f_{i_1}^{'}$, …, $(x_{i_{n_i^{'}}-1}^{'}, z_i^{'}) : f_{i_{n_i^{'}}}^{'}$, $(z_i, z_i^{'}) : P_{os_i}$, …, $\forall i \in [1, m]$, with $u_i = f_{i_1} \ldots f_{i_{n_i}}$ and $u_i^{'} = f_{i_1}^{'} \ldots f_{i_{n_i^{'}}}^{'}$, which are satisfied by inductive hypothesis. It follows that $\mathcal{J}_{\mathcal{F}_Q}$ satisfies also $s : \Theta(C, D, \{\langle u_1, P_{os_1}, u_1^{'}\rangle, \ldots,$
$\langle u_m, P_{os_m}, u_m^{'}\rangle\})$.

The remaining cases require similar reasoning and are therefore dismissed. ■

Proof (Proposition 4)

The proof can be obtained by induction on the structure of the concept C. Suppose that $\mathcal{F}_Q$ is clash-free and that $\mathcal{J}_{\mathcal{F}_Q}$ satisfies $s : C$.

Suppose $C = A$ (i.e., a concept name), then $s : A \in \mathcal{F}_Q$ by definition of $\mathcal{J}_{\mathcal{F}_Q}$, since $\mathcal{J}_{\mathcal{F}_Q}$ satisfies $s : A$.
If C is of the form $C_1 \sqcap C_2$. Then, $\mathcal{J}_{\mathcal{F}_Q}$ satisfies $s : C_1 \sqcap C_2$ iff $\mathcal{J}_{\mathcal{F}_Q}$ satisfies both $s : C_1$ and $s : C_2$. By inductive hypothesis, this is the case iff $s : C_1 \in \mathcal{F}_Q$ and $s : C_2 \in \mathcal{F}_Q$. Since $s : C_1 \sqcap C_2 \in \mathcal{G}_\mathcal{V}$, we have $s : C_1 \sqcap C_2 \in \mathcal{F}_Q$, since otherwise rule $\mathbf{C_1}$ would be applicable.
If C is of the form $P_r(u_1, \ldots, u_n)$. Then, $\mathcal{J}_{\mathcal{F}_Q}$ satisfies $s : P_r(u_1, \ldots, u_n)$ iff $\mathcal{J}_{\mathcal{F}_Q}$ satisfies $(s, z_1) : u_1, \ldots, (s, z_n) : u_n, (z_1, \ldots, z_n) : P'_r$ for some $z_1, \ldots, z_n$ where $P'_r(u_1, \ldots, u_n)$ entails $P_r(u_1, \ldots, u_n)$. By inductive hypothesis, this is the case iff $(s, z_1) : u_1 \in \mathcal{F}_Q, \ldots, (s, z_n) : u_n \in \mathcal{F}_Q, (z_1, \ldots, z_n) : P'_r \in \mathcal{F}_Q$. Since $s : P_r(u_1, \ldots, u_n) \in \mathcal{G}_\mathcal{V}$, we have $s : P_r(u_1, \ldots, u_n) \in \mathcal{F}_Q$ since otherwise rule $\mathbf{C_2}$ would be applicable.
If C is of the form $\{a\}$. Recall that the calculus starts with a pair $\{x' : Q\}.\{x' : V\}$. We make the following remarks: **(1)** By inspecting all the rules of the calculus we see that any individual t occuring in a constraint $t : C'$ in $\mathcal{G}_\mathcal{V}$ occurs also in $\mathcal{F}_Q$. **(2)** By analyzing the rules we see that if a constant (i.e., an abstract individual name) a occurs in $\mathcal{F}_Q$, then $\mathcal{F}_Q$ contains a constraint of the form $t : \{a\}$. Hence, if $\mathcal{J}_{\mathcal{F}_Q}$ satisfies $s : \{a\}$, then by definition $s^\mathcal{J} = a$. The remark (1) leads to the fact that a occurs also in $\mathcal{F}_Q$, and the remark (2) leads to the fact that $\mathcal{F}_Q$ contains $t : \{a\}$. It follows that $t^\mathcal{J} = a$ and then $a : \{a\}$ is in $\mathcal{F}_Q$.
If C is of the form
$\Theta(C, D, \{\langle u_1, P_{os_1}, u_1' \rangle, \ldots, \langle u_m, P_{os_m}, u_m' \rangle\})$, then $\mathcal{J}_{\mathcal{F}_Q}$ satisfies $s : \Theta(C, D, \{\langle u_1, P_{os_1}, u_1' \rangle, \ldots, \langle u_m, P_{os_m}, u_m' \rangle\})$ iff it satisfies $s : C$, $t : D$, $\ldots$, $f_{i_1}(s) = x_{i_1}$, $\ldots$, $f_{i_{n_i}}(x_{i_{n_i}-1}) = z_i$, $f'_{i_1}(t) = x'_{i_1}$, $\ldots$, $f'_{i_{n'_i}}(x'_{i_{n'_i}-1}) = z'_i$
for some t. By inductive hypothesis, this is the case iff each of these constraints belongs to $\mathcal{F}_Q$. Since $s : \Theta(C, D, \{\langle u_1, P_{os_1}, u_1' \rangle, \ldots, \langle u_m, P_{os_m}, u_m' \rangle\}) \in \mathcal{G}_\mathcal{V}$, we have $s : \Theta(C, D, \{\langle u_1, P_{os_1}, u_1' \rangle, \ldots, \langle u_m, P_{os_m}, u_m' \rangle\}) \in \mathcal{F}_Q$ since otherwise $\mathbf{C_3}$ would be applicable.
If C is of the form $\exists R.C$, then $\mathcal{J}_{\mathcal{F}_Q}$ satisfies $s : \exists R.C$ iff it satisfies $sRt, t : C$ for some t. This is the case iff $sRt \in \mathcal{F}_Q$, and $t : C \in \mathcal{F}_Q$. Since $s : \exists R.C \in \mathcal{G}_\mathcal{V}$, we have $s : \exists R.C \in \mathcal{F}_Q$ since otherwise $\mathbf{C_5}$ would be applicable.
If C is of the form $\top$, then $s : \top \in \mathcal{F}_Q$ since otherwise $\mathbf{C_4}$ would be applicable.

A similar reasoning is required for the other cases.

■

Proof (Proposition 5)

The Proof follows from the following arguments.

The size of Q is finite. Since $\mathcal{S}$ is acyclic and the size of Q is finite, the number of direct successors of an individual s is finite. When one of the generating rules $\mathbf{D_2}$, $\mathbf{D_3}$, $\mathbf{D_4}$, $\mathbf{S_4}$, $\mathbf{S_5}$, $\mathbf{S_6}$ is applied to a constraint of the form $s : C$, the number of variables or concrete individual names that are generated is less or equal to the size of C, and if a constraint of the form $y : C'$ is generated then C' is always a strict sub-expression of C. All rules but $\rightarrow_\forall$ are not applied twice on the same constraint. The rule $\rightarrow_\forall$ is never applied to an individual s more than the number of direct successors of s. The schema $\mathcal{S}$ is acyclic and contains a finite number of axioms. Hence, the number of times of application of the rule $\mathbf{S_4}$ is finite.

Consider the rules that alter the goal. As the size of V is finite the number of application of the rules $\mathbf{G_1}$ and $\mathbf{G_3}$ is finite. As the chain leading from $\{x' : Q\}$ to $\mathcal{F}_Q$ is finite, it follows that $\mathcal{F}_Q$ contains a finite number of constraints, and then the number of application of the rules $\mathbf{G_2}$ and $\mathbf{G_4}$ is finite. ■

Proof (Theorem 1)

If $\mathcal{F}_Q$ contains a clash, then $\mathcal{F}_Q$ is unsatisfiable. As $x' : Q$ is in $\mathcal{F}_Q$, according to the Proposition 2, $x' : Q$ is unsatisfiable. This means that Q is unsatisfiable and an unsatisfiable concept is subsumed by any concept.

We have seen that $Q \dot{\preceq}_{\mathcal{S}} V$ iff $x' : Q \models_{\mathcal{S}} x' : V$. (Proposition 1)

" $\Rightarrow$ "
Let $\mathcal{F}_Q$ be a clash-free constraint system and $x' : Q \models_{\mathcal{S}} x' : V$. According to the Corollary 2, we have $\mathcal{F}_Q \models_{\mathcal{S}} o : V$. Let $\mathcal{J}_{\mathcal{F}_Q}$ be the canonical interpretation of $\mathcal{F}_Q$. It follows from the Proposition 3 that $\mathcal{J}_{\mathcal{F}_Q}$ is a model of $\mathcal{F}_Q$. In this case it satisfies $o : V$. We have supposed $o : V \in \mathcal{G}_{\mathcal{V}}$. In this case, according to the Proposition 4 we have $o : V$ in $\mathcal{F}_Q$.

" $\Leftarrow$ "
If $o : V$ is in $\mathcal{F}_Q$, then $\mathcal{F}_Q \models_{\mathcal{S}} o : V$. According to the Corollary 2 we have $x' : Q \models_{\mathcal{S}} x' : V$. It follows from the Proposition 1 that $Q \dot{\preceq}_{\mathcal{S}} V$.

■

Proof (Proposition 6)

Any constant in the pair $\mathcal{F}_Q.\mathcal{G}_{\mathcal{V}}$ must appear in Q. The number of variables introduced by decomposition rules is bounded by $|Q|$. The number of variables introduced by schema rules is bounded by $|V|$. ■

Proof (Theorem 2)

The propagation rules can be devided in two categories: (1) rules that add constraints and (2) rules that reduce the number of variables (i.e., $\mathbf{D_6}$). The number of application of the rule that reduces the number of variables is finite and bounded by the size of Q. The number of application of the other decomposition rules is also finite and bounded by the size of Q. The number of application of the goal rules is finite and bounded by the size of V. The number of application of composition rules is finite and bounded by the size of V. The number of application of schema rules is finite and bounded by $(|Q| + |V|).|S|$ ■