

## An extended Relational Algebra on Abstract Objects for summarizing answers to queries

**Robert Demolombe**

*ONERA Toulouse*

*Robert.Demolombe@cert.fr*

---

**Abstract.** Answers to queries in terms of abstract objects are defined in the logical framework of first order predicate calculus. A partial algebraic characterisation of the supremum and of the infimum of abstract answers is given in an extended Relational Algebra of the Cylindric Algebra kind. Then, the form of queries is restricted in order to be able to compute answers without the cylindrification operator. For these restricted queries we give a technique to compute an upper bound and a lower bound of abstract answers using only the operators of the standard Relational Algebra.

**Keywords:** Relational Algebra, Cylindric Algebra, First Order Logic, Abstraction

### 1. Introduction

Let's consider, for instance, a database that contains statistical information about weather in the main cities of North America during the year 2000. This information is formally represented with the predicate  $weather(x, y, z)$ , where  $x$ ,  $y$  and  $z$  respectively denote a city, a day in the year, and the weather. For instance, the fact that it was cold in Boston on January 20th is represented by:  $weather(Boston, 20/01, cold)$ . Queries are expressed by formulas of a first order predicate calculus language. For instance the query: *in which cities was the weather on December 25th the same as in Miami?*, is expressed by:  $\exists y(weather(Miami, 25/12, y) \wedge weather(x, 25/12, y))$ , and the query: *in which cities and on which days was the weather warm and not raining?*, is expressed by:  $weather(x, y, warm) \wedge \neg weather(x, y, raining)$ . The answer to the second query may be, for instance, the relation R in Table 1.

A more concise answer can be obtained if the elements in the definition domains are represented by abstract objects<sup>1 2</sup>. For instance, days can be abstracted into months, and months can be abstracted into

---

Address for correspondence: 2 Avenue E. Belin, 31055 Toulouse Cedex 4, France

<sup>1</sup>This approach has been previously investigated by T. Ellman in [3] and by T. Imielinski in [5].

<sup>2</sup>Here the term "object" does not refer to Object Oriented languages.

seasons. In the same way, cities can be abstracted into states, and states can be abstracted into countries. The definitions of these abstract objects are, for instance:

January={01/01,02/01,...,31/01}, ..., December={01/12,,02/12,...,31/12},  
 Winter={January,February,March}, ..., Autumn={September,October,...}.  
 Florida={Miami,Orlando,...}, Georgia={Atlanta,Albany,...},  
 Massachusetts={Boston,Providence,...}, Ontario={Toronto,Kingston,...},  
 USA={Florida,Georgia,...}, Canada={Ontario,Manitoba,...}.

Then, the answer given in relation R can be represented at the first abstraction level by the relation R1, and at the second abstraction level by the relation R2.

Table 1. Abstracted answers

R		R1		R2	
Miami	10/01	Florida	October	USA	Autumn
Miami	11/02	Florida	November	Canada	Autumn
Albany	10/02	Georgia	October	...	...
Albany	10/04	Massachusetts	October		
Boston	10/02	Ontario	October		
Toronto	10/03	...	...		
...	...				

However, it may be that the weather is not warm in all the cities in the USA in Autumn, nor in all the cities in Ontario in October. So, we have to make clear what is the meaning of the presence or absence of a tuple in these abstract relations.

We have considered here two different possible meanings. For instance, the meaning of the tuple  $\langle \text{Florida}, \text{October} \rangle$  can be that for every city  $x$  in Florida, and for every day  $y$  in October, the weather is warm in  $x$  on day  $y$ , or it can be that there exists at least one city  $x$  in Florida and one day  $y$  in October such that the weather is warm in  $x$  on day  $y$ .

If we formally see abstract objects (at any abstract level) as unary predicates that take their values from the domain of the database relations, these two definitions can be expressed respectively by:

$$\forall x \forall y (Florida(x) \wedge October(y) \rightarrow weather(x, y, warm)), \text{ and}$$

$$\exists x \exists y (Florida(x) \wedge October(y) \wedge weather(x, y, warm)).$$

The fact that Miami is abstracted into Florida, and Florida is abstracted into USA, could be formally represented by:  $Florida(Miami)$ , and  $USA(Florida)$ . However, with this formalisation we would have unary predicate symbols, such as  $Florida$ , as arguments of unary predicates, as in  $USA(Florida)$ . That means that we would have a higher order logic. To avoid this complication, and to stay within first order logic, we have preferred to represent these facts by the formulas:  $\forall x (Florida(x) \rightarrow USA(x))$ ,  $Florida(Miami)$ . For homogeneity, the fact  $Florida(Miami)$  could equivalently be represented by:  $\forall x (x = Miami \rightarrow Florida(x))$ .

The aim of this paper<sup>3</sup> is first to give a formal definition of these kinds of abstract answers in formal logic. Then, we define an extended Relational Algebra to compute upper bounds and lower bounds of the abstract answers (section 2). However, this algebra, which is a sort of Cylindric Algebra is too expensive

<sup>3</sup>A preliminary and restricted version of this work has been presented in [2].

from a computational point of view.

For that reason we restrict it to the operators of standard Relational Algebra (section 3). The final result is that standard Relational Algebra can be used to compute upper bounds and lower bounds of abstract answers at different abstraction levels.

## 2. Formal logical framework

The formal definition of knowledge representation at the different abstract levels is given below.

### Definition 2.1. (Knowledge representation at the abstraction level 0)

#### Language.

Let  $L$  be a first order predicate calculus language without function symbols and with the logical connectives: negation, denoted by  $\neg$ , and disjunction, denoted by  $\vee$ , and the existential quantifier, denoted by  $\exists$ . It is assumed that variables symbols are of the form:  $x, x_1, x_2, \dots, x_n, \dots$ . The other logical connectives and quantifier are defined from disjunction, negation and the universal quantifier as usual. Conjunction is denoted by  $\wedge$ , implication is denoted by  $\rightarrow$ , equivalence is denoted by  $\leftrightarrow$ , and the universal quantifier is denoted by  $\forall$ .

We call  $L_0$  an extension of the language  $L$  with a new unary predicate symbol  $d$ .

The predicate  $d$  is intended to represent the database definition domain.

#### Database.

A “database”  $DB$  is a set of range restricted definite Horn clauses<sup>4</sup>. The set of closed world assumption axioms for the database  $DB$  is denoted by  $CWA$  [10], and  $CDB$  is used to denote the set of sentences  $DB \cup CWA$ .  $D$  is used to denote the set of constant symbols that occur in  $DB$ .

Let us call  $DB_0$  the set of sentences  $DB_0 = DB \cup (\bigcup_{c \in D} d(c))$ . The set of sentences  $CWA_0 \cup DB_0$  is denoted by  $CDB_0$ , where  $CWA_0$  is the set of closed world assumption axioms for  $DB_0$ .

### Definition 2.2. (Knowledge representation at the abstraction level $l$ )

#### Language.

The language  $L_l$  is defined from the language  $L_{l-1}$  by adding to the predicate symbols in  $L_{l-1}$  the new unary predicate symbols:  $a_1^l, \dots, a_n^l$ .

The predicates  $a_1^l, \dots, a_n^l$  are intended to represent the abstract objects at the level  $l$ .

#### Database.

The domain  $D_l$  at the abstraction level  $l$  is the set of predicate symbols:  $a_1^l, \dots, a_n^l$ .

The database  $DB_l$  is defined from the database  $DB_{l-1}$  by adding to  $DB_{l-1}$  the following sentences:

- $\forall x (d(x) \leftrightarrow a_1^l(x) \vee \dots \vee a_n^l(x))$ , where  $a_1^l, \dots, a_n^l$  are the elements of  $D_l$
- for every element  $a_i^l$  in  $D_l$ :
  - a set of sentences of the form:  $a_i^l(c_1), \dots, a_i^l(c_{n_i})$ , where  $c_1, \dots, c_{n_i}$  are in  $D$
  - $\exists x a_i^l(x)$
- for every pairs of distinct elements  $a_i^l$  and  $a_j^l$  in  $D_l$ :  $\neg \exists x (a_i^l(x) \wedge a_j^l(x))$

<sup>4</sup>Range restricted Horn clauses are clauses with exactly one positive literal, such that each variable that occurs in the positive literal also occurs in at least one negative literal [7].

- for every element  $a_j^{l-1}$  in  $D_{l-1}$ :  $\forall x(a_j^{l-1}(x) \rightarrow a_k^l(x))$ , where  $a_k^l$  is some element in  $D_l$

The set of sentences  $CWA_l \cup DB_l$ , where  $CWA_l$  is the set of closed world assumption axioms for  $DB_l$ , is called  $CDB_l$ . It is assumed that  $CDB_l$  is consistent.

From an intuitive point of view, the elements of  $D_l$  define a partition on the elements of  $D$ . Also, if  $a_k^l(a_j^{l-1})$  is used to denote the formula  $\forall x(a_j^{l-1}(x) \rightarrow a_k^l(x))$ ,  $a_k^l(a_j^{l-1})$  can be interpreted as the fact that  $a_j^{l-1}$  is in the extension of  $a_k^l$  at the abstraction level  $l - 1$  (notice that  $a_k^l(a_j^{l-1})$  is not a formula in  $L_l$ ). From this point of view, we can intuitively say that the elements of  $D_l$  define a partition on the elements of  $D_{l-1}$ .

We shall use the following notation.

$\vec{x} = \langle x_{i_1}, \dots, x_{i_n} \rangle$ , where  $i_1 < i_2 < \dots < i_n$  (intuitively if  $x_{i_j}$  is in  $\vec{x}$ ,  $i_j$  is the index of the variable symbol and  $j$  is its rank in the tuple  $\vec{x}$ );

$\vec{c} = \langle c_1, \dots, c_n \rangle$ , where every  $c_i$  is in  $D$ ;

$a^l = \langle a_1^l, \dots, a_n^l \rangle$ , where every  $a_i^l$  is in  $D_l$ ;

$a^l(\vec{x}) = a_1^l(x_{i_1}) \wedge \dots \wedge a_n^l(x_{i_n})$ ;

$\exists \vec{x} = \exists x_{i_1} \dots \exists x_{i_n}$ ;  $\forall \vec{x} = \forall x_{i_1} \dots \forall x_{i_n}$ .

**Definition 2.3. (Supremum of  $\phi$  at the abstraction level  $l$ )**

Let  $\phi(\vec{x})$  be a formula of  $L$  whose free variables are  $\vec{x}$ . The supremum  $sup^l(\phi(\vec{x}))$  of  $\phi$  at the abstraction level  $l$  is a set of tuples  $a^l$  in  $(D_l)^n$  such that:

$$sup^l(\phi(\vec{x})) \stackrel{\text{def}}{=} \{a^l : \vdash CDB_l \rightarrow \exists \vec{x}(a^l(\vec{x}) \wedge \phi(\vec{x}))\}$$

**Definition 2.4. (Infimum of  $\phi$  at the abstraction level  $l$ )**

Let  $\phi(\vec{x})$  be a formula of  $L$  whose free variables are  $\vec{x}$ . The infimum  $inf^l(\phi(\vec{x}))$  of  $\phi$  at the abstraction level  $l$  is a set of tuples  $a^l$  in  $(D_l)^n$  such that:

$$inf^l(\phi(\vec{x})) \stackrel{\text{def}}{=} \{a^l : \vdash CDB_l \rightarrow \forall \vec{x}(a^l(\vec{x}) \rightarrow \phi(\vec{x}))\}$$

We shall use the notation:

$$S^l(\vec{x}) \stackrel{\text{def}}{=} \bigvee_{a^l \in sup^l(\phi(\vec{x}))} a^l(\vec{x})$$

$$I^l(\vec{x}) \stackrel{\text{def}}{=} \bigvee_{a^l \in inf^l(\phi(\vec{x}))} a^l(\vec{x})$$

Intuitively  $S^l(\vec{x})$  and  $I^l(\vec{x})$  characterize the extensions of the supremum and of the infimum at the abstract level 0.

**Theorem 2.1.** We have the following properties:

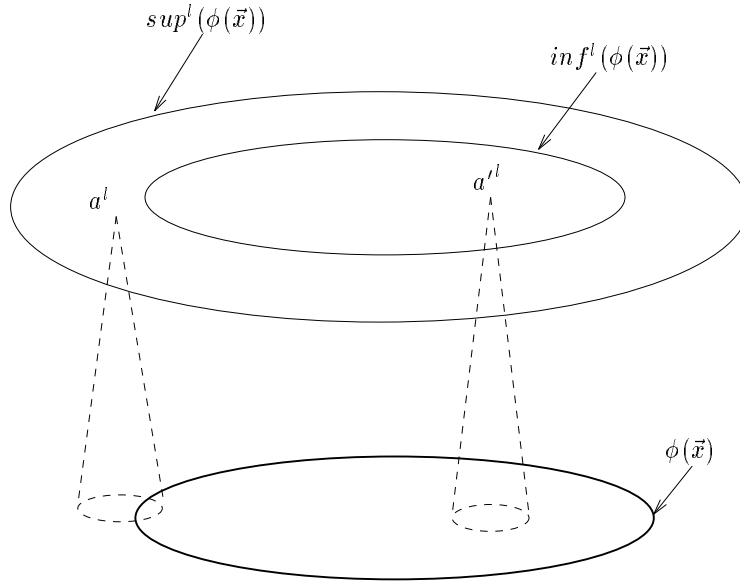
$$\vdash CDB_l \rightarrow \forall \vec{x}(I^l(\vec{x}) \rightarrow \phi(\vec{x}))$$

$$\vdash CDB_l \rightarrow \forall \vec{x}(\phi(\vec{x}) \rightarrow S^l(\vec{x}))$$

**Proof:**

Let  $\{c_1, \dots, c_m\}$  be the set of elements in  $D$ . From the definition of  $DB_0$  the sentences  $d(c_1), \dots, d(c_m)$  are in  $DB_0$ , and the Domain Closure Axiom in  $CWA_0$  is of the form:  $\forall x(x = c_1 \vee \dots \vee x = c_m)$ . Then, we have:  $\vdash CDB_0 \rightarrow \forall x d(x)$ .

The set of constant symbols is the same in  $DB_l$  as in  $DB_0$ , and the Domain Closure Axiom is the same in  $CDB_l$  as in  $CDB_0$ . Then, we also have:  $\vdash CDB_l \rightarrow \forall x d(x)$ .

Figure 1. Infimum and supremum at the abstraction level  $l$ .

In  $CDB_l$  we have:  $\forall x (d(x) \leftrightarrow a_1^l(x) \vee \dots \vee a_n^l(x))$  (where  $a_1^l, \dots, a_n^l$  are the elements in  $D_l$ ), which can be expressed in the form  $\forall x (d(x) \leftrightarrow \bigvee_{a^l \in D_l} a^l(x))$ . Then, we have: (1)  $\vdash CDB_l \rightarrow \forall x (\bigvee_{a^l \in D_l} a^l(x))$ .

If it assumed that we have: (2)  $\not\vdash CDB_l \rightarrow \forall \vec{x} (\phi(\vec{x}) \rightarrow S^l(\vec{x}))$ , we can infer  $\vdash CDB_l \rightarrow \neg \forall \vec{x} (\phi(\vec{x}) \rightarrow S^l(\vec{x}))$ , because  $CDB_l$  is a complete theory, and we have: (3)  $\vdash CDB_l \rightarrow \exists \vec{x} (\phi(\vec{x}) \wedge \neg S^l(\vec{x}))$ . From (1) and (3), we have:  $\vdash CDB_l \rightarrow \exists \vec{x} ((\bigvee_{a^l \in D_l} a^l(\vec{x})) \wedge \phi(\vec{x}) \wedge \neg S^l(\vec{x}))$ .

From the definition of  $S^l(\vec{x})$ ,  $\neg S^l(\vec{x})$  is logically equivalent to:  $\bigwedge_{a^l \in \text{sup}^l(\phi(\vec{x}))} \neg a^l(\vec{x})$ , and  $(\bigvee_{a^l \in D_l} a^l(\vec{x})) \wedge (\bigwedge_{a^l \in \text{sup}^l(\phi(\vec{x}))} \neg a^l(\vec{x}))$  is logically equivalent to:  $\bigvee_{a^l \in D_l \text{ and } a^l \notin \text{sup}^l(\phi(\vec{x}))} a^l(\vec{x})$ .

Therefore, we have: (4)  $\vdash CDB_l \rightarrow \exists \vec{x} (\bigvee_{a^l \in D_l \text{ and } a^l \notin \text{sup}^l(\phi(\vec{x}))} a^l(\vec{x}) \wedge \phi(\vec{x}))$ , which contradicts the definition of  $\text{sup}^l(\phi(\vec{x}))$ , since all the  $a^l$  such that  $\vdash CDB_l \rightarrow \exists \vec{x} (a^l(\vec{x}) \wedge \phi(\vec{x}))$  are in  $\text{sup}^l(\phi(\vec{x}))$ .

Since (2) leads to a contradiction we have:  $\vdash CDB_l \rightarrow \forall \vec{x} (\phi(\vec{x}) \rightarrow a^l(\vec{x}))$ .

From the definition of  $\text{inf}^l(\phi(\vec{x}))$ , if  $a^l$  is in  $\text{inf}^l(\phi(\vec{x}))$  we have:  $\vdash CDB_l \rightarrow \forall \vec{x} (a^l(\vec{x}) \rightarrow \phi(\vec{x}))$ . Therefore, we have:  $\vdash CDB_l \rightarrow \forall \vec{x} ((\bigvee_{a^l \in \text{inf}^l(\phi(\vec{x}))} a^l(\vec{x})) \rightarrow \phi(\vec{x}))$ , and, from the definition of  $I^l(\vec{x})$ , we have:  $\vdash CDB_l \rightarrow \forall \vec{x} (I^l(\vec{x}) \rightarrow \phi(\vec{x}))$ . □

**Definition 2.5. (Upper bound of  $\phi$  at the abstraction level  $l$ )**

Let  $\phi(\vec{x})$  be a formula of  $L$  whose free variables are  $\vec{x}$ . An upper bound  $\text{upp}^l(\phi(\vec{x}))$  at the abstraction level  $l$  is any subset of  $(D_l)^n$  such that  $\text{sup}^l(\phi(\vec{x})) \subseteq \text{upp}^l(\phi(\vec{x}))$ .

**Definition 2.6. (Lower bound of  $\phi$  at the abstraction level  $l$ )**

Let  $\phi(\vec{x})$  be a formula of  $L$  whose free variables are  $\vec{x}$ . A lower bound  $\text{low}^l(\phi(\vec{x}))$  at the abstraction level  $l$  is any subset of  $(D_l)^n$  such that  $\text{low}^l(\phi(\vec{x})) \subseteq \text{inf}^l(\phi(\vec{x}))$ .

**Definition 2.7. (Algebra at the abstraction level  $l$ )**

An algebra at the abstraction level  $l$  is defined on subsets of  $(D_l)^n$ . The operators  $\cup$  (union),  $-$  (difference),  $\times$  (cartesian product) and  $S_\sigma$  (selection) are defined as usual in standard Relational Algebra [11].

For convenience, the projection operator  $\Pi_i$  is not defined here as usual. If  $E$  is a subset of  $(D_l)^n$ , we have:

$$\Pi_i(E) = \{ \langle a_1^l, \dots, a_{i-1}^l, a_{i+1}^l, \dots, a_n^l \rangle : \text{there exists } a_i^l \text{ in } D_l \text{ such that } \langle a_1^l, \dots, a_{i-1}^l, a_i^l, a_{i+1}^l, \dots, a_n^l \rangle \in E \}$$

The only difference with the standard definition is that  $i$  denotes the component of the tuples that is removed by the projection operator (in the standard definition the index, or the indices, define the components that are preserved by the projection operator).

If we have  $I = \langle i_1, \dots, i_p \rangle$ ,  $\Pi_I(E)$  is used to denote  $\Pi_{i_1}(\dots(\Pi_{i_p}(E))\dots)$ .

We also have in the algebra a permutation operator which is denoted by  $P_{I/J}$ , where  $J = \langle j_1, \dots, j_p \rangle$  is a permutation of  $I$  such that  $j_1 < j_2 < \dots < j_p$ . If  $E$  is a subset of  $(D_l)^p$ , we have:

$$P_{I/J}(E) = \{ \langle a_1^l, \dots, a_p^l \rangle : \text{there exists } \langle a_{k_1}^l, \dots, a_{k_p}^l \rangle \text{ in } E \text{ such that for every } s \text{ in } [1, p], a_s^l \text{ is } a_{k_m}^l \text{ such that } i_m \text{ in } I \text{ is equal to } j_s \text{ in } J \}$$

Finally, in algebraic formulas we allow operands of the form  $(D_l)^n$ .

Comment: it is worth noting that by allowing operands of the form  $(D_l)^n$  we give to this algebra the same expressive power as Cylindric Algebra. Indeed, for a subset  $E$  of  $(D_l)^n$ , the cylindrication operator  $C_i$  can be defined as follows [6]:

$$C_i(E) = \{ \langle a_1^l, \dots, b_i^l, \dots, a_n^l \rangle : b_i^l \in D_l \text{ and there exists } a_i^l \in D_l \text{ such that } \langle a_1^l, \dots, a_i^l, \dots, a_n^l \rangle \in E \}$$

Then, we can easily check that we have:  $C_i(E) = P_{I/J}(\Pi_i(E) \times D_l)$ , where  $P_{I/J}$  is the appropriate permutation.

For instance, if we have  $I = \langle i_1, i_2, i_3, i_4 \rangle = \langle 3, 5, 2, 6 \rangle$  and  $J = \langle j_1, j_2, j_3, j_4 \rangle = \langle 2, 3, 5, 6 \rangle$ , we have  $j_1 = i_3$ ,  $j_2 = i_1$ ,  $j_3 = i_2$  and  $j_4 = i_4$ . The result of the permutation  $P_{I/J}$ , when it is applied to the tuple  $\langle a_1^l, a_2^l, a_3^l, a_4^l \rangle$ , is  $\langle a_3^l, a_1^l, a_2^l, a_4^l \rangle$ .

We shall use the following notation. Let  $\phi(\vec{x})$  and  $\psi(\vec{y})$  be formulas of  $L$  whose free variables respectively are  $\vec{x} = \langle x_{i_1}, \dots, x_{i_p} \rangle$  and  $\vec{y} = \langle x_{k_1}, \dots, x_{k_r} \rangle$ . If  $a^l$  is an instance of  $\vec{x}$ , we have  $a^l = \langle a_{i_1}^l, \dots, a_{i_p}^l \rangle$ , and if  $a^l$  is an instance of  $\vec{y}$ , we have  $a^l = \langle a_{k_1}^l, \dots, a_{k_r}^l \rangle$ .

Let  $\vec{z}$  be the tuple of variables  $\vec{z} = \langle x_{j_1}, \dots, x_{j_q} \rangle$  such that  $\{x_{j_1}, \dots, x_{j_q}\} = \{x_{i_1}, \dots, x_{i_p}\} \cup \{x_{k_1}, \dots, x_{k_r}\}$ , and  $j_1 < j_2 < \dots < j_q$ , we shall use the notation  $a^l(\vec{x}) = a_{i_1}^l(x_{i_1}) \wedge \dots \wedge a_{i_p}^l(x_{i_p})$ , and  $a^l(\vec{y}) = a_{k_1}^l(x_{k_1}) \wedge \dots \wedge a_{k_r}^l(x_{k_r})$ .

We also shall use the notation:

$$I_1 = \langle i_1, \dots, i_p \rangle, K_1 = \langle k_1, \dots, k_r \rangle, J = \langle j_1, \dots, j_q \rangle,$$

$I_2 = \langle i'_1, \dots, i'_{q-p} \rangle$  such that  $\{i'_1, \dots, i'_{q-p}\} = \{j_1, \dots, j_q\} - \{i_1, \dots, i_p\}$  and  $i'_1 < i'_2 < \dots < i'_{q-p}$  (intuitively  $I_2$  is the tuple of the indices of the variable symbols that occur in  $\vec{z}$  and do not occur in  $\vec{x}$ ),

$K_2 = \langle k'_1, \dots, k'_{q-r} \rangle$  such that  $\{k'_1, \dots, k'_{q-r}\} = \{j_1, \dots, j_q\} - \{k_1, \dots, k_r\}$  and  $k'_1 < k'_2 < \dots < k'_{q-r}$  (intuitively  $K_2$  is the tuple of the indices of the variable symbols that occur in  $\vec{z}$  and do not occur in  $\vec{y}$ ),

$$I = \langle I_1, I_2 \rangle \text{ and } K = \langle K_1, K_2 \rangle.$$

For instance, if we have:  $\vec{x} = \langle x_3, x_5 \rangle$  and  $\vec{y} = \langle x_2, x_3, x_6 \rangle$ , we have  $\vec{z} = \langle x_2, x_3, x_5, x_6 \rangle$ , and

$$\begin{aligned}
I_1 &= \langle i_1, i_2 \rangle = \langle 3, 5 \rangle, K_1 = \langle k_1, k_2, k_3 \rangle = \langle 2, 3, 6 \rangle, \\
J &= \langle j_1, j_2, j_3, j_4 \rangle = \langle 2, 3, 5, 6 \rangle, \\
I_2 &= \langle i'_1, i'_2 \rangle = \langle 2, 6 \rangle, K_2 = \langle k'_1 \rangle = \langle 5 \rangle, \\
I &= \langle 3, 5, 2, 6 \rangle \text{ and } K = \langle 2, 3, 5, 6 \rangle.
\end{aligned}$$

**Theorem 2.2.** The supremum and infimum of formulas at the abstraction level  $l$  satisfy the following properties.

- (1)  $sup^l(\phi(\vec{x}) \vee \psi(\vec{y})) = P_{I/J}(sup^l(\phi(\vec{x})) \times (D_l)^r) \cup P_{K/J}(sup^l(\psi(\vec{y})) \times (D_l)^p)$
- (2)  $P_{I/J}(inf^l(\phi(\vec{x})) \times (D_l)^r) \cup P_{K/J}(inf^l(\psi(\vec{y})) \times (D_l)^p) \subseteq inf^l(\phi(\vec{x}) \vee \psi(\vec{y}))$
- (3)  $sup^l(\neg\phi(\vec{x})) = (D_l)^p - inf^l(\phi(\vec{x}))$
- (4)  $inf^l(\neg\phi(\vec{x})) = (D_l)^p - sup^l(\phi(\vec{x}))$
- (5)  $sup^l(\exists x_{i_m} \phi(\vec{x})) = \Pi_m(sup^l(\phi(\vec{x})))$
- (6)  $\Pi_m(inf^l(\phi(\vec{x}))) \subseteq inf^l(\exists x_{i_m} \phi(\vec{x}))$

Let  $p$  be an  $n$ -ary predicate symbol in  $L$ . We use the following notation  $R_l^+ = sup^l(p(x_1, \dots, x_n))$  and  $R_l^- = inf^l(p(x_1, \dots, x_n))$ . Let  $p(t_1, \dots, t_n)$  be an atomic formula in  $L$ . We have:

- (7)  $sup^l(p(t_1, \dots, t_n)) = P_{I/J}(\Pi_{I'}(S_\sigma(R_l^+)))$
- (8)  $inf^l(p(t_1, \dots, t_n)) = P_{I/J}(\Pi_{I'}(S_\sigma(R_l^-)))$

where  $\sigma, I', I$  and  $J$  are defined as follows.

The selection condition  $\sigma$  is a conjunction of atomic conditions. If  $t_s$  is the constant symbol  $c_{i_s}$ , then the condition  $(s = c_{i_s})$  is in  $\sigma$ . If  $t_s$  is the variable symbol  $x_{i_s}$ , and there exists  $s'$  such that  $s' < s$  and  $t_s = t_{s'}$  and there is no  $s''$  such that  $s' < s''$  and  $s'' < s$  and  $t_s = t_{s''}$ , then  $s' = s$  is in  $\sigma$ . There is no other condition in  $\sigma$ .

$I$  and  $I'$  are the index tuples  $I = \langle i_{s_1}, \dots, i_{s_p} \rangle$  and  $I' = \langle s'_1, \dots, s'_{n-p} \rangle$  such that  $\{s'_1, \dots, s'_{n-p}\} = \{1, \dots, n\} - \{s_1, \dots, s_p\}$  and for  $r$  in  $[1, p]$  we have:  $t_{s_r}$  is a variable symbol  $x_{i_{s_r}}$  and there is no  $s'$  such that  $s' < s$  and  $t_{s_r} = t_{s'}$ .  $J$  is the index tuple  $J = \langle j_1, \dots, j_p \rangle$  such that  $J$  is a permutation of  $I$ , and we have  $j_1 < j_2 < \dots < j_p$ .

### Proof:

Case of (1).

We have  $sup^l(\phi(\vec{x}) \vee \psi(\vec{y})) = \{a^l : \vdash CDB_l \rightarrow \exists \vec{z}(a^l(\vec{z}) \wedge (\phi(\vec{x}) \vee \psi(\vec{y})))\}$ . Since  $\exists \vec{z}(a^l(\vec{z}) \wedge (\phi(\vec{x}) \vee \psi(\vec{y})))$  is logically equivalent to  $\exists \vec{z}(a^l(\vec{z}) \wedge \phi(\vec{x})) \vee \exists \vec{z}(a^l(\vec{z}) \wedge \psi(\vec{y}))$ , we have:  $sup^l(\phi(\vec{x}) \vee \psi(\vec{y})) = \{a^l : \vdash CDB_l \rightarrow \exists \vec{z}(a^l(\vec{z}) \wedge \phi(\vec{x}))\} \cup \{a^l : \vdash CDB_l \rightarrow \exists \vec{z}(a^l(\vec{z}) \wedge \psi(\vec{y}))\}$ .

Let  $\vec{x}^j$  be  $\vec{x}^j = \langle x_{i'_1}, \dots, x_{i'_{q-p}} \rangle$  (where  $\langle i'_1, \dots, i'_{q-p} \rangle = I_2$ ). Then,  $\exists \vec{z}(a^l(\vec{z}) \wedge \phi(\vec{x}))$  is logically equivalent to  $\exists \vec{x}(a^l(\vec{x}) \wedge \phi(\vec{x})) \wedge \exists \vec{x}^j a^l(\vec{x}^j)$ . From the definition of  $DB_l$ , for every  $a^l_i$  in  $D_l$  we have  $\exists x a^l_i(x)$  in  $DB_l$ , then  $\exists x a^l_i(x)$  holds in the context of  $CDB_l$ , and in that context  $\exists \vec{x}(a^l(\vec{x}) \wedge \phi(\vec{x})) \wedge \exists \vec{x}^j a^l(\vec{x}^j)$  is equivalent to  $\exists \vec{x}(a^l(\vec{x}) \wedge \phi(\vec{x}))$ .

Then, we have  $\{a^l : \vdash CDB_l \rightarrow \exists \vec{z}(a^l(\vec{z}) \wedge \phi(\vec{x}))\} = \{a^l : \vdash CDB_l \rightarrow \exists \vec{x}(a^l(\vec{x}) \wedge \phi(\vec{x})) \text{ and } \langle a^l_{i'_1}, \dots, a^l_{i'_{q-p}} \rangle \in (D_l)^{q-p}\}$ , because here  $a^l$  is an instance of  $\vec{z}$ .

Since we have:  $sup^l(\phi(\vec{x})) = \{ \langle a^l_{i_1}, \dots, a^l_{i_p} \rangle : \vdash CDB_l \rightarrow \exists \vec{x}(a^l_{i_1}(x_{i_1}) \wedge \dots \wedge a^l_{i_p}(x_{i_p}) \wedge \phi(\vec{x})) \}$ , we have:  $\{a^l : \vdash CDB_l \rightarrow \exists \vec{z}(a^l(\vec{z}) \wedge \phi(\vec{x}))\} = P_{I/J}(sup^l(\phi(\vec{x})) \times (D_l)^{q-p})$ .

In the same way we have:  $\{a^l : \vdash CDB_l \rightarrow \exists \vec{z}(a^l(\vec{z}) \wedge \psi(\vec{y}))\} = P_{K/J}(sup^l(\psi(\vec{y})) \times (D_l)^{q-r})$ . Then, we have (1)  $sup^l(\phi(\vec{x}) \vee \psi(\vec{y})) = P_{I/J}(sup^l(\phi(\vec{x})) \times (D_l)^r) \cup P_{K/J}(sup^l(\psi(\vec{y})) \times (D_l)^p)$ .

Case of (2).

We have  $\text{inf}^l(\phi(\vec{x}) \vee \psi(\vec{y})) = \{a^l : \vdash CDB_l \rightarrow \forall \vec{z}(a^l(\vec{z}) \rightarrow (\phi(\vec{x}) \vee \psi(\vec{y})))\}$ . If some  $a^l$  in  $(D_l)^p$  satisfies either the property  $\vdash CDB_l \rightarrow \forall \vec{z}(a^l(\vec{z}) \rightarrow \phi(\vec{x}))$  or the property  $\vdash CDB_l \rightarrow \forall \vec{z}(a^l(\vec{z}) \rightarrow \psi(\vec{y}))$ , then it satisfies the property  $\vdash CDB_l \rightarrow \forall \vec{z}(a^l(\vec{z}) \rightarrow (\phi(\vec{x}) \vee \psi(\vec{y})))$  (notice that in general the implication in the other way round does not hold).

Then we have:

$$\{a^l : \vdash CDB_l \rightarrow \forall \vec{z}(a^l(\vec{z}) \rightarrow \phi(\vec{x}))\} \cup \{a^l : \vdash CDB_l \rightarrow \forall \vec{z}(a^l(\vec{z}) \rightarrow \psi(\vec{y}))\} \subseteq \{a^l : \vdash CDB_l \rightarrow \forall \vec{z}(a^l(\vec{z}) \rightarrow (\phi(\vec{x}) \vee \psi(\vec{y})))\}.$$

$\forall \vec{z}(a^l(\vec{z}) \rightarrow \phi(\vec{x}))$  is logically equivalent to  $\forall \vec{x}(\exists \vec{x}' a^l(\vec{x}') \wedge a^l(\vec{x}) \rightarrow \phi(\vec{x}))$ , because the variables that occur in  $\vec{x}'$  does not occur in  $\vec{x}$ . Since  $\exists \vec{x}' a^l(\vec{x}')$  holds in the context of  $CDB_l$ , in that context  $\forall \vec{z}(a^l(\vec{z}) \rightarrow \phi(\vec{x}))$  is equivalent to  $\forall \vec{x}(a^l(\vec{x}) \rightarrow \phi(\vec{x}))$ . Then, with a similar technique as for the proof of (1) we can prove that:

$$\{a^l : \vdash CDB_l \rightarrow \forall \vec{z}(a^l(\vec{z}) \rightarrow \phi(\vec{x}))\} = P_{I/J}(\text{inf}^l(\phi(\vec{x})) \times (D_l)^{q-p}).$$

It can be proved in the same way that:

$$\{a^l : \vdash CDB_l \rightarrow \forall \vec{z}(a^l(\vec{z}) \rightarrow \psi(\vec{y}))\} = P_{K/J}(\text{inf}^l(\psi(\vec{y})) \times (D_l)^{q-r}).$$

Then, we have: (2)  $P_{I/J}(\text{inf}^l(\phi(\vec{x})) \times (D_l)^r) \cup P_{K/J}(\text{inf}^l(\psi(\vec{y})) \times (D_l)^p) \subseteq \text{inf}^l(\phi(\vec{x}) \vee \psi(\vec{y})).$

Case of (3).

We have:  $\text{sup}^l(\neg\phi(\vec{x})) = \{a^l : \vdash CDB_l \rightarrow \exists \vec{x}(a^l(\vec{x}) \wedge \neg\phi(\vec{x}))\}$ .

Since  $CDB_l$  is a complete and consistent theory, we have:  $\vdash CDB_l \rightarrow \exists \vec{x}(a^l(\vec{x}) \wedge \neg\phi(\vec{x}))$  iff  $\not\vdash CDB_l \rightarrow \neg\exists \vec{x}(a^l(\vec{x}) \wedge \neg\phi(\vec{x}))$ . Then, we have:

$\text{sup}^l(\neg\phi(\vec{x})) = \{a^l : \not\vdash CDB_l \rightarrow \forall \vec{x}(a^l(\vec{x}) \rightarrow \phi(\vec{x}))\}$ , and

$\text{sup}^l(\neg\phi(\vec{x})) = \{a^l : a^l \in (D_l)^p \text{ and } a^l \notin \text{inf}^l(\phi(\vec{x}))\}$ .

Therefore, we have: (3)  $\text{sup}^l(\neg\phi(\vec{x})) = (D_l)^p - \text{inf}^l(\phi(\vec{x})).$

Case of (4).

The proof of (4) is very similar to the proof of (3).

Case of (5).

Let us now use the following notation.

$$a^l = \langle a^l_{i_1}, \dots, a^l_{i_{m-1}}, a^l_{i_{m+1}}, \dots, a^l_{i_p} \rangle$$

$$\vec{x}' = \langle x_{i_1}, \dots, x_{i_{m-1}}, x_{i_{m+1}}, \dots, x_{i_p} \rangle$$

We have:  $\text{sup}^l(\exists x_{i_m} \phi(\vec{x})) = \{a^l : \vdash CDB_l \rightarrow \exists \vec{x}'(a^l(\vec{x}') \wedge \exists x_{i_m} \phi(\vec{x}))\}$ .

It can be proven that  $\exists x_{i_m} \phi(\vec{x})$  holds in the context of  $CDB_l$  iff there exists some  $a^l_{i_m}$  in  $D_l$  such that  $\exists x_{i_m}(a^l_{i_m}(x_{i_m}) \wedge \phi(\vec{x}))$  holds in that context (the proof is very similar to the proof of Theorem 1). Then, we have:  $\text{sup}^l(\exists x_{i_m} \phi(\vec{x})) = \{a^l : \text{there exists } a^l_{i_m} \text{ in } D_l \text{ such that } \vdash CDB_l \rightarrow \exists \vec{x}'(a^l(\vec{x}') \wedge \exists x_{i_m}(a^l_{i_m}(x_{i_m}) \wedge \phi(\vec{x}))\}$ .

The property  $\vdash CDB_l \rightarrow \exists \vec{x}'(a^l(\vec{x}') \wedge \exists x_{i_m}(a^l_{i_m}(x_{i_m}) \wedge \phi(\vec{x})))$  is equivalent to  $\vdash CDB_l \rightarrow \exists \vec{x}(a^l(\vec{x}) \wedge \phi(\vec{x}))$ , which is equivalent to  $a^l \in \text{sup}^l(\phi(\vec{x}))$ .

Then, we have:  $\text{sup}^l(\exists x_{i_m} \phi(\vec{x})) = \{a^l : \text{there exists } a^l_{i_m} \text{ in } D_l \text{ such that } a^l \in \text{sup}^l(\phi(\vec{x}))\}$ . Therefore, we have: (5)  $\text{sup}^l(\exists x_{i_m} \phi(\vec{x})) = \Pi_m(\text{sup}^l(\phi(\vec{x}))).$

Case of (6).

We have;  $\Pi_m(\text{inf}^l(\phi(\vec{x}))) = \{a^l : \text{there exists } a_{i_m}^l \text{ in } D_l \text{ such that } a^l \in \text{inf}^l(\phi(\vec{x}))\}$ .

From the definition of  $\text{inf}^l$ ,  $a^l \in \text{inf}^l(\phi(\vec{x}))$  is equivalent to  $\vdash CDB_l \rightarrow \forall \vec{x}(a^l(\vec{x}) \rightarrow \phi(\vec{x}))$ , which can be expressed in the form:  $\vdash CDB_l \rightarrow \forall \vec{x} \forall x_{i_m}(a^l(\vec{x}) \wedge a_{i_m}^l(x_{i_m}) \rightarrow \phi(\vec{x}))$ . And this last property implies (but it is not equivalent to)  $\vdash CDB_l \rightarrow \forall \vec{x}(a^l(\vec{x}) \wedge (\exists x_{i_m} a_{i_m}^l(x_{i_m})) \rightarrow \exists x_{i_m} \phi(\vec{x}))$ .

Since  $\exists x_{i_m} a_{i_m}^l(x_{i_m})$  is in  $CDB_l$ , this is equivalent to  $\vdash CDB_l \rightarrow \forall \vec{x}(a^l(\vec{x}) \rightarrow \exists x_{i_m} \phi(\vec{x}))$ , which is equivalent to  $a^l \in \text{inf}^l(\exists x_{i_m} \phi(\vec{x}))$ .

Finally, the property: there exists  $a_{i_m}^l$  in  $D_l$  such that  $a^l \in \text{inf}^l(\phi(\vec{x}))$  implies the property  $a^l \in \text{inf}^l(\exists x_{i_m} \phi(\vec{x}))$ . Therefore, we have: (6)  $\Pi_m(\text{inf}^l(\phi(\vec{x}))) \subseteq \text{inf}^l(\exists x_{i_m} \phi(\vec{x}))$

Case of (7).

According to the definition of  $I$  and  $I'$ ,  $\{i_{s_1}, \dots, i_{s_p}\}$  are the indices of the variables that occur in  $p(t_1, \dots, t_n)$ , and  $\{s_1, \dots, s_p\}$  are the indices of the first occurrence (going from 1 to  $n$ ) of these variables in  $p(t_1, \dots, t_n)$ , and  $\{s'_1, \dots, s'_{n-p}\}$  are the indices of the terms in  $p(t_1, \dots, t_n)$  that are constant symbols or variable symbols that have already occurred (going from 1 to  $n$ ) in  $p(t_1, \dots, t_n)$ .

Let  $p(t'_1, \dots, t'_n)$  be the atomic formula defined from  $p(t_1, \dots, t_n)$  as follows. For every  $s$  in  $[1, n]$ , if  $s$  is such that  $i_s$  occurs in  $I$ , then  $t'_s$  is  $x_{i_s}$ , else  $t'_s$  is a variable symbol  $x_{j_s}$  which has no other occurrence in  $p(t'_1, \dots, t'_n)$ .

If  $\sigma$  is of the form  $\sigma_1 \wedge \dots \wedge \sigma_q$ , for each  $k$  in  $[1, q]$ ,  $\sigma'_k$  is defined from  $\sigma_k$  as follows. If  $\sigma_k$  is of the form  $s'_k = s_k$ , then  $\sigma'_k$  is  $x_{i_{s'_k}} = x_{i_{s_k}}$ , else, if  $\sigma_k$  is of the form  $s_k = c_{i_k}$ , then  $\sigma'_k$  is  $x_{j_{s_k}} = c_{i_k}$ , where  $x_{j_{s_k}}$  is the term  $t'_{s_k}$  in  $p(t'_1, \dots, t'_n)$ . Let us call  $\sigma'$  the formula  $\sigma'_1 \wedge \dots \wedge \sigma'_q$ .

Let us call  $\vec{x}'$  the tuple  $\langle x_{j_{s'_1}}, \dots, x_{j_{s'_{n-p}}} \rangle$  such that  $\vec{x}'$  is the tuple of variable symbols that occur in  $p(t'_1, \dots, t'_n)$  and does not occur in  $p(t_1, \dots, t_n)$ .

Then the formula  $\exists \vec{x}'(p(t'_1, \dots, t'_n) \wedge \sigma')$  is logically equivalent to  $p(t_1, \dots, t_p)$ .

Therefore we have  $\text{sup}^l(p(t_1, \dots, t_p)) = \text{sup}^l(\exists \vec{x}'(p(t'_1, \dots, t'_n) \wedge \sigma'))$  and we have (7)  $\text{sup}^l(p(t_1, \dots, t_n)) = P_{I/J}(\Pi_{I'}(S_\sigma(R_l^+)))$ .

Case of (8).

The proof is the same as the proof of (7). □

**Corollary 2.1.** The supremum and infimum of formulas with conjunctions and universal quantifiers at the abstraction level  $l$  satisfy the following properties.

- (9)  $\text{sup}^l(\phi(\vec{x}) \wedge \psi(\vec{y})) \subseteq P_{I/J}(\text{sup}^l(\phi(\vec{x})) \times (D_l)^r) \cap P_{K/J}(\text{sup}^l(\psi(\vec{y})) \times (D_l)^p)$
- (10)  $\text{inf}^l(\phi(\vec{x}) \wedge \psi(\vec{y})) = P_{I/J}(\text{inf}^l(\phi(\vec{x})) \times (D_l)^r) \cap P_{K/J}(\text{inf}^l(\psi(\vec{y})) \times (D_l)^p)$
- (11)  $\text{sup}^l(\forall x_{i_m} \phi(\vec{x})) \subseteq (D_l)^{p-1} - \Pi_m((D_l)^p - \text{sup}^l(\phi(\vec{x})))$
- (12)  $\text{inf}^l(\forall x_{i_m} \phi(\vec{x})) = (D_l)^{p-1} - \Pi_m((D_l)^p - \text{inf}^l(\phi(\vec{x})))$

**Proof:**

These properties can be easily proven by rewriting the  $\wedge$  operator and the  $\forall$  quantifier in terms of the operators  $\vee$  and  $\neg$  and of the quantifier  $\exists$ . Then, we just have to use the results of theorem 2.2. □

The reason why in (2) and (6) we have an inclusion instead of an equality can be intuitively understood with simple counter examples.

Let's consider, for example, the formulas  $\phi(x_1) = \text{weather}(\text{Boston}, x_1, \text{raining})$  and  $\psi(x_1) = \text{weather}(\text{Boston}, x_1, \text{snowing})$ , and  $\phi(x_1) \vee \psi(x_1)$ , which defines on which days it is either raining or snowing in Boston. If January belongs to  $\text{inf}^1(\phi(x_1) \vee \psi(x_1))$ , that is, if on every day in January it is either raining or snowing in Boston, it is not necessarily the case that either on every day in January it is raining in Boston, or on every day in January it is snowing in Boston. That means that January does not necessarily belong to  $\text{inf}^1(\phi(x_1))$  nor to  $\text{inf}^1(\psi(x_1))$ . That is why we do not have an equality in (2).

Let's consider now the formula  $\text{weather}(x_1, x_2, \text{foggy})$ , which defines in which cities and on which days it is foggy. If the state Massachusetts belongs to  $\text{inf}^1(\exists x_2 \text{weather}(x_1, x_2, \text{foggy}))$ , that is, if for every city in Massachusetts there exists a day on which it is foggy, it is not necessarily the case that there exists a month such that for every city in Massachusetts and for every day in that month it is foggy in this city on that day. That means that there does not necessarily exist a month M such that  $\langle \text{Massachusetts}, M \rangle$  belongs to  $\text{inf}^1(\text{weather}(x_1, x_2, \text{foggy}))$ , and Massachusetts does not necessarily belong to  $\Pi_1(\text{inf}^1(\text{weather}(x_1, x_2, \text{foggy})))$ . That is why we do not have an equality in (6).

The notation used in the properties (7) and (8) can be illustrated by the following example.

Let us consider the atomic formula  $p(x_3, c_1, x_2, x_3)$ , we have  $I = \langle i_{s_1}, i_{s_2} \rangle = \langle 3, 2 \rangle$  and  $\langle s_1, s_2 \rangle = \langle 1, 3 \rangle$ , because the first occurrences of the variable symbols  $x_3$  and  $x_2$  respectively are the terms  $t_1$  and  $t_3$ . Then, we have  $I' = \langle 2, 4 \rangle$ .

The selection condition  $\sigma$  is  $\sigma_1 \wedge \sigma_2$ , where  $\sigma_1$  is  $(2 = c_1)$  because the term  $t_2$  is the constant symbol  $c_1$ , and  $\sigma_2$  is  $(1 = 4)$  because we have  $t_1 = t_4 = x_3$ .

Then,  $p(t'_1, \dots, t'_4)$  is, for instance,  $p(x_3, x_4, x_2, x_5)$ , and  $\sigma'_1$  is  $(x_4 = c_1)$  and  $\sigma'_2$  is  $(x_3 = x_5)$ . We also have  $\vec{x}' = \langle x_4, x_5 \rangle$ .

The formula  $\exists \vec{x}' (p(t'_1, t'_2, t'_3, t'_4) \wedge \sigma')$  is in this example  $\exists x_3 \exists x_4 (p(x_3, x_4, x_2, x_5) \wedge (x_4 = c_1) \wedge (x_3 = x_4))$  and we can easily check that it is logically equivalent to  $p(x_3, c_1, x_2, x_3)$ .

Since  $J$  is  $\langle 2, 3 \rangle$  we finally have  $\text{sup}^1(p(x_3, c_1, x_2, x_3)) = P_{\langle 3, 2 \rangle / \langle 2, 3 \rangle} (\Pi_{2,4} (S_{(2=c_1) \wedge (1=4)} (R_i^+)))$ .

**Theorem 2.3.** Let  $\theta(\vec{x})$  be a formula of  $L$ . An upper bound  $\text{upp}^l(\theta(\vec{x}))$  of  $\theta(\vec{x})$  and a lower bound  $\text{low}^l(\theta(\vec{x}))$  of  $\theta(\vec{x})$  can be computed from the supremum and the infimum of atomic sentences that occur in  $\theta(\vec{x})$  with the following formulas.

$$\begin{aligned} \text{upp}^l(\phi(\vec{x}) \vee \psi(\vec{y})) &= P_{I/J}(\text{upp}^l(\phi(\vec{x})) \times (D_i)^r) \cup P_{K/J}(\text{upp}^l(\psi(\vec{y})) \times (D_i)^p) \\ \text{low}^l(\phi(\vec{x}) \vee \psi(\vec{y})) &= P_{I/J}(\text{low}^l(\phi(\vec{x})) \times (D_i)^r) \cup P_{K/J}(\text{low}^l(\psi(\vec{y})) \times (D_i)^p) \\ \text{upp}^l(\neg\phi(\vec{x})) &= (D_i)^p - \text{low}^l(\phi(\vec{x})) \\ \text{low}^l(\neg\phi(\vec{x})) &= (D_i)^p - \text{upp}^l(\phi(\vec{x})) \\ \text{upp}^l(\exists x_{i_m} \phi(\vec{x})) &= \Pi_m(\text{upp}^l(\phi(\vec{x}))) \\ \text{low}^l(\exists x_{i_m} \phi(\vec{x})) &= \Pi_m(\text{low}^l(\phi(\vec{x}))) \\ \text{upp}^l(p(t_1, \dots, t_n)) &= \text{sup}^l(p(t_1, \dots, t_n)) \\ \text{low}^l(p(t_1, \dots, t_n)) &= \text{inf}^l(p(t_1, \dots, t_n)). \end{aligned}$$

**Proof:**

Theorem 2.3 can be easily proven from theorem 2.2 by induction on the length of formulas.  $\square$

**Corollary 2.2.** An upper bound and a lower bound of a formula  $\theta(\vec{x})$  with conjunction or universal quantifier can be computed with the following formulas.

$$\begin{aligned}
upp^l(\phi(\vec{x}) \wedge \psi(\vec{y})) &= P_{I/J}(upp^l(\phi(\vec{x})) \times (D_l)^r) \cap P_{K/J}(upp^l(\psi(\vec{y})) \times (D_l)^p) \\
low^l(\phi(\vec{x}) \wedge \psi(\vec{y})) &= P_{I/J}(low^l(\phi(\vec{x})) \times (D_l)^r) \cap P_{K/J}(low^l(\psi(\vec{y})) \times (D_l)^p) \\
upp^l(\forall x_{i_m} \phi(\vec{x})) &= (D_l)^{p-1} - \Pi_m((D_l)^p - upp^l(\phi(\vec{x}))) \\
low^l(\forall x_{i_m} \phi(\vec{x})) &= (D_l)^{p-1} - \Pi_m((D_l)^p - low^l(\phi(\vec{x}))).
\end{aligned}$$

**Proof:**

Corollary 2.2 is a direct consequence of theorem 2.3.  $\square$

**2.1. Extension to many sorted logic**

The results presented in this section can be extended without any theoretical difficulties to many sorted first order languages.

The extension would require defining one sort for each predicate argument, and restricting the set of formulas in the language  $L$  such that predicate arguments in a formula that are occupied by the same variable symbol have the same sort. A consequence of this restriction is that one sort can be assigned to each variable symbol in the context of a given formula.

Then, the language  $L_0$  could be defined as an extension of the language  $L$  with as many unary predicate symbols  $d_1(x), \dots, d_n(x)$  as there are distinct sorts.

For each sort  $i$ , the elements of the domain  $D_{l_i}$  at the abstraction level  $l_i$  could be denoted by  $a_1^{l_i}, \dots, a_p^{l_i}$ .

The abstraction level for a formula with  $k$  free variables could be defined by the tuple:  $l = \langle \langle i_1, l_1 \rangle, \dots, \langle i_k, l_k \rangle \rangle$ , where  $i_1, \dots, i_k$  are sorts indices and  $l_1, \dots, l_k$  are the corresponding abstraction levels.

If the definition domains for each sort at the level 0 are denoted by  $D^1, \dots, D^n$ , the domain  $D_l$  at the abstraction level  $l$  could be defined by  $D_l = D_{l_1}^{i_1} \times \dots \times D_{l_k}^{i_k}$ , and a tuple  $a^l$  in  $D_l$  could be defined by:  $a^l = \langle a_1^{i_1, l_1}, \dots, a_k^{i_k, l_k} \rangle$ , where each  $a_j^{i_j, l_j}$  is in  $D_{l_j}^{i_j}$ .

All the results presented in Theorem 1 and Theorem 2 would be the same, except that cartesian products of the form  $(D_l)^p$  should be replaced by cartesian products of the appropriate  $D_{l_j}^{i_j}$ s.

**3. Restriction to standard Relational Algebra**

The results presented in theorem 2.3 show that to compute upper bounds and lower bounds we have to compute cartesian products where some operands are of the form  $(D_l)^n$ . That would be extremely expensive. To have reasonable costs we have to restrict the algebra by removing this kind of operation which intuitively corresponds to the cylindrification operator. However, it has been shown (see [8, 1]) that it is not possible to define a corresponding restriction in the first order predicate calculus. For that reason we have defined in this section a subset  $L_A$  of the first order predicate calculus whose translation in the Relational Algebra does not require the cylindrification operator.

For convenience we give as a first step the definition of an extended join operator.

**Definition 3.1. (Extended join operator)**

For the definition we shall use the following notation.

$$\begin{aligned}
\vec{x} &= \langle x_{i_1}, \dots, x_{i_p} \rangle \\
\vec{y} &= \langle x_{k_1}, \dots, x_{k_r} \rangle
\end{aligned}$$

$$I_1 = \langle i_1, \dots, i_p \rangle$$

$$K_1 = \langle k_1, \dots, k_r \rangle$$

$J_1 = \langle j_1, \dots, j_q \rangle$ , such that  $\{j_1, \dots, j_q\} = \{i_1, \dots, i_p\} \cup \{k_1, \dots, k_r\}$  and  $j_1 < j_2 < \dots < j_q$  (intuitively  $J_1$  is the tuple of indices of variable symbols that occur either in  $\vec{x}$  or in  $\vec{y}$ ).

$K' = \langle l'_1, \dots, l'_n \rangle$ , such that  $l'_m$  is in  $K'$  iff  $l'_m = l_m + p$  and there exists some  $i_s$  in  $I_1$  such that  $k_{l'_m} = i_s$  (intuitively  $x_{k_{l'_m}}$  is a variable symbol that occurs in  $\vec{y}$  and also occurs in  $\vec{x}$ ,  $l_m$  is the rank of  $x_{k_{l'_m}}$  in the tuple  $\vec{y}$ , and  $l'_m$  is the rank of  $x_{k_{l'_m}}$  in the tuple  $\langle x_{i_1}, \dots, x_{i_p}, x_{k_1}, \dots, x_{k_r} \rangle$ ).

$J_2 = \langle i_1, \dots, j_p, k'_1, \dots, k'_{r-n} \rangle$ , where  $\langle k'_1, \dots, k'_{r-n} \rangle$  is a tuple obtained from  $K_1$  by removing all the components  $k_s$  such that there exists some  $i_t$  in  $I_1$  such that  $k_s = i_t$  (intuitively  $\langle k'_1, \dots, k'_{r-n} \rangle$  is the tuple of indices of the variable symbols that occur in  $\vec{y}$  and do not occur in  $\vec{x}$ ).

Let us denote by  $\sigma$  the conjunction of all the atomic conditions of the form  $(s = t')$ , where  $t' = p + t$ , such that  $i_s$  in  $I_1$  is equal to  $k_t$  in  $K_1$  (intuitively the variable symbol  $x_{i_s}$  in  $\vec{x}$  and the variable symbol  $x_{k_t}$  in  $\vec{y}$  are identical).

The extended join operator, denoted by  $\otimes$ , is defined by:

$$\text{bound}^l(\phi(\vec{x})) \otimes \text{bound}^l(\psi(\vec{y})) \stackrel{\text{def}}{=} P_{J_2/J_1}(\Pi_{K'}(S_\sigma(\text{bound}^l(\phi(\vec{x})) \times \text{bound}^l(\psi(\vec{y}))))))$$

where  $\text{bound}^l$  may be either  $\text{upp}^l$  or  $\text{low}^l$ .

For example, if we have  $\vec{x} = \langle x_2, x_5 \rangle$  and  $\vec{y} = \langle x_3, x_5, x_7 \rangle$ , we have  $I_1 = \langle 2, 5 \rangle$ ,  $K_1 = \langle 3, 5, 7 \rangle$  and  $J_1 = \langle 2, 3, 5, 7 \rangle$ . We also have  $k_{l'_m} = i_s$  for  $l'_m = 2$  and  $s = 2$ . Since  $p = 2$ , we have  $l'_m = l_m + p = 2 + 2 = 4$ . Then, we have  $K' = \langle 4 \rangle$ . Since 5 occurs in  $K_1$  and also occurs in  $J_1$  we have  $\langle k'_1, k'_2 \rangle = \langle 3, 7 \rangle$ , and  $J_2 = \langle 2, 5, 3, 7 \rangle$ . We also have  $i_s = k_t$  for  $s = 2$  and  $t = 2$ , then we have  $t' = t + p = 2 + 2 = 4$  and  $\sigma = (s = t') = (2 = 4)$ .

Finally, for that example, we have

$$P_{J_2/J_1}(\Pi_{K'}(S_\sigma(\text{bound}^l(\phi(\vec{x})) \times \text{bound}^l(\psi(\vec{y})))))) = P_{\langle 2,5,3,7 \rangle / \langle 2,3,5,7 \rangle}(\Pi_4(S_{(2=4)}(\text{bound}^l(\phi(\vec{x})) \times \text{bound}^l(\psi(\vec{y}))))))$$

Comment. Depending on the variables  $\vec{x}$  and  $\vec{y}$ , the operator  $\otimes$  simplifies into either a standard cartesian product and a permutation, or an intersection, or a composition of several standard joins and a permutation.

In the case where  $\vec{x} = \vec{y}$  we have  $J_2 = I_1$  and we can easily check that  $\Pi_{K'}(S_\sigma(\text{bound}^l(\phi(\vec{x})) \times \text{bound}^l(\psi(\vec{x}))))$  is equivalent to  $\text{bound}^l(\phi(\vec{x})) \cap \text{bound}^l(\psi(\vec{x}))$ . Then, we have:  $\text{bound}^l(\phi(\vec{x})) \otimes \text{bound}^l(\psi(\vec{x})) = P_{J_2/J_1}(\text{bound}^l(\phi(\vec{x})) \cap \text{bound}^l(\psi(\vec{x})))$ .

In the case where  $\vec{x}$  and  $\vec{y}$  have no common variable symbol we have  $K' = \emptyset$  and  $\sigma = \emptyset$ . Then, we have:  $\text{bound}^l(\phi(\vec{x})) \otimes \text{bound}^l(\psi(\vec{y})) = P_{J_2/J_1}(\text{bound}^l(\phi(\vec{x})) \times \text{bound}^l(\psi(\vec{y})))$ .

Let us define in our context the standard join operator, which is denoted by  $\circ_{(s=t)}$ , as follows. If  $E$  and  $F$  respectively are subsets of  $(D_l)^p$  and  $(D_l)^r$ , the standard join of  $E$  and  $F$ , for the condition  $(s = t)$  is defined by:

$$E \circ_{(s=t)} F = \{ \langle a_1^l, \dots, a_p^l, b_1^l, \dots, b_r^l \rangle : \langle a_1^l, \dots, a_p^l \rangle \in E \text{ and } \langle b_1^l, \dots, b_r^l \rangle \in F \text{ and } a_s^l = b_t^l \}$$

In the case where the sets of variable symbols in  $\vec{x}$  and  $\vec{y}$  overlap, neither  $K'$  nor  $\sigma$  are empty. Then,  $\sigma$  is of the form  $\sigma' \wedge (s = t)$ , and we can easily check that we have  $\text{bound}^l(\phi(\vec{x})) \otimes \text{bound}^l(\psi(\vec{y})) = P_{J_2/J_1}(\Pi_{K'}(S_{\sigma'}(\text{bound}^l(\phi(\vec{x})) \circ_{(s=t)} \text{bound}^l(\psi(\vec{y}))))))$ .

### Definition 3.2. (Restricted language $L_A$ )

The language  $L_A$  is a restriction of  $L$  which is completely defined by the following rules.

$var(\phi)$  is used to denote the set of free variables in the formula  $\phi$ .

- Atomic formulas are in  $L_A$ .
- If  $\phi$  and  $\psi$  are in  $L_A$ , and  $var(\phi) = var(\psi)$ ,  $(\phi \vee \psi)$  is in  $L_A$ .
- If  $\phi$  and  $\psi$  are in  $L_A$ ,  $(\phi \wedge \psi)$  is in  $L_A$ .
- If  $\phi$  and  $\psi$  are in  $L_A$ , and  $var(\psi) \subseteq var(\phi)$ ,  $(\phi \wedge \neg\psi)$  is in  $L_A$ .
- If  $\phi$  is in  $L_A$ , and  $x_i \in var(\phi)$ ,  $(\exists x_i \phi)$  is in  $L_A$ .
- If  $\phi$  and  $\psi$  are in  $L_A$ , and  $x_i \in var(\phi)$ , and  $x_i \in var(\psi)$ ,  $((\exists x_i \phi) \wedge \forall x_i (\phi \rightarrow \psi))$  is in  $L_A$ .

**Theorem 3.1.** For any formula in  $L_A$ ,  $upp^l$  and  $low^l$  can be computed from the  $sup^l$  and the  $inf^l$  of the atomic formulas that occur in it using the following formulas.

- (1)  $upp^l(\phi \vee \psi) = upp^l(\phi) \cup upp^l(\psi)$
- (2)  $low^l(\phi \vee \psi) = low^l(\phi) \cup low^l(\psi)$
- (3)  $upp^l(\phi \wedge \psi) = upp^l(\phi) \otimes upp^l(\psi)$
- (4)  $low^l(\phi \wedge \psi) = low^l(\phi) \otimes low^l(\psi)$
- (5)  $upp^l(\phi \wedge \neg\psi) = upp^l(\phi) - (low^l(\phi) \otimes low^l(\psi))$
- (6)  $low^l(\phi \wedge \neg\psi) = low^l(\phi) - (upp^l(\phi) \otimes upp^l(\psi))$
- (7)  $upp^l(\exists x_{i_m} \phi) = \Pi_m(upp^l(\phi))$
- (8)  $low^l(\exists x_{i_m} \phi) = \Pi_m(low^l(\phi))$
- (9)  $upp^l((\exists x_{i_m} \phi) \wedge \forall x_{i_m} (\phi \rightarrow \psi)) = ((\Pi_m(upp^l(\phi))) \otimes (\Pi_m(upp^l(\psi)))) - \Pi_m((low^l(\phi) \otimes \Pi_m(low^l(\psi))) - ((\Pi_m(upp^l(\phi)) \otimes upp^l(\psi))))$
- (10)  $low^l((\exists x_{i_m} \phi) \wedge \forall x_{i_m} (\phi \rightarrow \psi)) = ((\Pi_m(low^l(\phi))) \otimes (\Pi_m(low^l(\psi)))) - \Pi_m((upp^l(\phi) \otimes \Pi_m(upp^l(\psi))) - ((\Pi_m(low^l(\phi)) \otimes low^l(\psi))))$
- (11)  $upp^l(p(t_1, \dots, t_n)) = sup^l(p(t_1, \dots, t_n))$
- (12)  $low^l(p(t_1, \dots, t_n)) = inf^l(p(t_1, \dots, t_n))$

**Proof:**

Case of (1) and (2).

Since  $\phi \vee \psi$  is in  $L_A$ , we have  $var(\phi) = var(\psi)$ . Therefore, in the formula given in theorem 2.3 we have  $r = 0$  and  $I = J = K$ . Then, we directly have (1) and (2).

Case of (3) and (4).

Let us assume that  $\phi \wedge \psi$  is of the form  $\phi(\vec{x}) \wedge \psi(\vec{y})$ .

Let  $\vec{y}'$  be a tuple of variable symbols obtained from  $\vec{y}$  by replacing the variable symbols  $x_{k_{l_m}}$  such that  $k_{l_m}$  is in  $K$ , by another variable symbol  $x_{k'_{l_m}}$  which has no other occurrence in  $\vec{y}'$  nor in  $\vec{x}$ .

Let  $\sigma'$  obtained from  $\sigma$  by replacing the atomic conditions of the form  $(s = t')$  by  $(x_{i_s} = x_{k'_{l_t}})$ , where  $t' = p + t$ .

Then,  $\phi(\vec{x}) \wedge \psi(\vec{y})$  is logically equivalent to  $\exists \vec{y}'(\phi(\vec{x}) \wedge \psi(\vec{y}') \wedge \sigma')$ . Therefore, according to the definition of the operator  $\otimes$  we have (3) and (4).

Case of (5) and (6).

The formula  $\phi \wedge \neg\psi$  is logically equivalent to  $\phi \wedge \neg(\phi \wedge \psi)$ . Since  $\phi \wedge \neg\psi$  is in  $L_A$  we have  $var(\psi) \subseteq var(\phi)$ . Then, we have  $var(\phi) = var(\neg(\phi \wedge \psi))$ , and from Corollary 2 we have  $upp^l(\phi \wedge \neg(\phi \wedge \psi)) = upp^l(\phi) \cap upp^l(\neg(\phi \wedge \psi))$ .

From Theorem 3, we have  $upp^l(\neg(\phi \wedge \psi)) = (D_l)^p - low^l(\phi \wedge \psi)$ , where  $p = |var(\phi \wedge \psi)|$ . Then, we have  $upp^l(\phi) \cap (D_L)^p = upp^l(\phi)$ , and we have  $upp^l(\phi \wedge \neg\psi) = upp^l(\phi) - low^l(\phi \wedge \psi)$ . Then, we have (5).

(6) can be proven in the same way.

Case of (7) and (8).

These cases have been already proven in the theorem 2.3.

Case of (9) and (10).

It can be easily checked that the formula  $(\exists x_{i_m} \phi) \wedge \forall x_{i_m} (\phi \rightarrow \psi)$  is logically equivalent to  $(\exists x_{i_m} \phi) \wedge \neg \exists x_{i_m} (\phi \wedge \neg\psi)$ , which is logically equivalent to  $(\exists x_{i_m} \phi) \wedge (\exists x_{i_m} \psi) \wedge \neg \exists x_{i_m} ((\phi \wedge (\exists x_{i_m} \psi)) \wedge \neg((\exists x_{i_m} \phi) \wedge \psi))$ .

Let us call  $\Phi_1$  the formula  $(\exists x_{i_m} \phi) \wedge (\exists x_{i_m} \psi)$ , and  $\Phi_2$  the formula  $\exists x_{i_m} ((\phi \wedge (\exists x_{i_m} \psi)) \wedge \neg((\exists x_{i_m} \phi) \wedge \psi))$ .

We can easily check that we have  $var(\Phi_1) = var(\Phi_2) = (var(\phi) \cup var(\psi)) - \{x_{i_m}\}$ . Then, from theorem 2.3, we have  $upp^l(\Phi_1 \wedge \neg\Phi_2) = upp^l(\Phi_1) - low^l(\Phi_2)$ . From (3) and (7) we have  $upp^l(\Phi_1) = (\Pi_m(upp^l(\phi))) \otimes (\Pi_m(upp^l(\psi)))$ .

Let us call  $\Psi_1$  the formula  $\phi \wedge (\exists x_{i_m} \psi)$  and  $\Psi_2$  the formula  $(\exists x_{i_m} \phi) \wedge g$ . We have  $var(\Psi_1) = var(\Psi_2) = (var(\phi) \cup var(\psi)) - \{x_{i_m}\}$ . Then, from theorem 2.3, we have  $low^l(\Psi_1 \wedge \neg\Psi_2) = low^l(\Psi_1) - upp^l(\Psi_2)$ .

Moreover, we have  $low^l(\Psi_1) = low^l(\phi) \otimes (\Pi_m(low^l(\psi)))$  and  $upp^l(\Psi_2) = (\Pi_m(upp^l(\phi))) \otimes upp^l(\psi)$ .

Since,  $low^l(\Phi_2) = \Pi_m(low^l(\Psi_1 \wedge \neg\Psi_2))$ , we finally have (9).

(10) can be proven in the same way.

□

## 4. Conclusion

We have presented an algebra to compute summarized answers to queries in terms of abstract objects. We have given a formal definition of an approximation defined by the supremum (*sup*) and the infimum (*inf*), and the theorems 2.2 and 2.3 give a partial characterization of the *sup* and *inf* of general formulas in terms of the *sup* and *inf* of atomic formulas at each abstraction level. This is mainly a theoretical result because the algebra involves the cylindrification operator.

Theorem 3.1 gives a practical method to compute an upper bound (*upp*) and a lower bound (*low*) using only the operators of the standard Relational Algebra. Then, the complexity of the computation of *upp* and *low* is the same as the complexity of the computation of answers to queries with a standard

relational database management system. It is worth noting that if there is no database update  $upp$  and  $low$  for atomic formulas can be computed only once for several queries.

An open question that could deserve further work is to analyse if it is possible to find a better approximation of the answers than those obtained from theorem 3.1.

In a private communication Churn-Jung Liao pointed out the close connection between the definition of  $sup$  and  $inf$  on the one hand and rough sets [9, 4] on the other. Indeed, the equivalence relation can be defined as the pairs of elements at the abstraction level  $l - 1$  which are abstracted by the same element at the abstraction level  $l$ . It would be interesting in the future to compare the two approaches in more detail.

## References

- [1] Demolombe, R.: Syntactical Characterization of a Subset of Domain Independent Formulas, *Journal of ACM*, 1982.
- [2] Demolombe, R.: Abstract objects to represent large answers to queries in a concise form, *Advances in Soft Computing. Flexible Query Answering Systems* (H. Larsen, J. Kacprzyk, S. Zadrozny, T. Andreassen, H. Christiansen, Eds.), Physica-Verlag, 2001.
- [3] Ellman, T.: Approximation and Abstraction Techniques for Generating Concise Answers to Database Queries, *Proc. of AAAI Spring Symposium*, 1995.
- [4] Green, J., Horne, N., Orłowska, E., Siemens, P.: A rough set model of information retrieval, *Fundamenta Informaticae*, **28**, 1996, 273–296.
- [5] Imielinski, T.: Domain abstraction and limited reasoning, *Proc. of the 10th IJCAI*, 1987.
- [6] Imielinski, T., Lipski, W.: *The Relational model of data and cylindric algebras*, Technical Report 446, Institute of Computer Science, Polish Academy of Science, 1981.
- [7] Nicolas, J.-M.: Logic for improving integrity checking in Relational Data Bases, *Acta Informatica*, **Vol 18**(Num 3), 1982.
- [8] Paola, R. A. D.: The recursive unsolvability of the decision problem for the class of Definite Formulas, *Journal of ACM*, **16**(2), 1969.
- [9] Pawlak, Z.: *Rough sets: theoretical aspects of reasoning about data*, Kluwer, 1991.
- [10] Reiter, R.: Towards a logical reconstruction of relational database theory, in: *On Conceptual Modelling: Perspectives from Artificial Intelligence, Databases and Programming Languages*, Springer Verlag, 1983.
- [11] Ullman, J. D.: *Principles of Database Systems*, Computer Science Press, 1980.