Semantic entailment of attribute-dependency formulas and their non-redundant bases

Radim Bělohlávek Vilém Vychodil

Dept. Computer Science, Palacký University, Tomkova 40, CZ-779 00, Olomouc, Czech Republic
Email: {radim.belohlavek, vilem.vychodil}@upol.cz

Abstract

AD-formulas are simple formulas representing particular dependencies in data tables. We study AD-formulas from the logical point of view. In particular, we present results regarding the relationship of AD-formulas to attribute implications, models and entailment of AD-formulas, non-redundant bases, and computation of non-redundant bases.

Keywords: attribute dependency, formal concept analysis, interior operators, logic in computer science, non-redundant basis

1. Introduction and preliminaries

The paper presents a contribution to reasoning about tabular data and, in particular, to formal concept analysis (FCA). FCA is a method of knowledge extraction from data tables describing relationships between objects and attributes, see [2, 4] for details. In a basic setting, a data table which enters FCA is represented by a triplet $\langle X,Y,I\rangle$ where X is a finite set of objects (table rows), Y is a finite set of attributes (columns), and $I \subseteq X \times Y$ is a binary relation between X and Y specifying whether an object x has an attribute y (in which case $\langle x,y\rangle \in I$) or not ($\langle x,y\rangle \notin I$). A (*formal*) *concept* in $\langle X,Y,I\rangle$ is a pair $\langle A,B\rangle$ of a set $A \subseteq X$ of objects (so-called *extent*) and a set $B \subseteq Y$ of attributes (so-called *intent*) such that A is the set of all objects which have all attributes from B, and B is the set of all attributes sharing all objects from A. Formal concepts can be partially ordered by putting $\langle A_1,B_1\rangle \leq \langle A_2,B_2\rangle$ iff $A_1 \subseteq A_2$ (or, equivalently, $B_2 \subseteq B_1$). A set $\mathcal{B}(X,Y,I)$ of all formal concepts in $\langle X,Y,I\rangle$ equipped with $\leq$ is called a *concept lattice* of $\langle X,Y,I\rangle$. $\mathcal{B}(X,Y,I)$ represents a collection of hierarchically ordered clusters extracted from the input data. Many applications of FCA can be found in [2] and in the references therein.

In [1], the authors introduced so-called attribute dependency formulas (AD-formulas). The aim is the following. It is very often the case that the user has some further information, additional to $\langle X,Y,I\rangle$. An AD-formula, e.g. "green color" $\sqsubseteq$ "cold-blooded" $\sqcup$ "warm-blooded", is an example of such an information expressing a relative importance of attributes (whether an animal is green is less important than whether it is cold-/warm-blooded). Then, for a given set T of AD-formulas, one can consider the collection $\mathcal{B}_T(X,Y,I)$ of concepts from $\mathcal{B}(X,Y,I)$ which respect T. The main benefits are the following: $\mathcal{B}_T(X,Y,I)$ is smaller and thus better comprehensible for a user than $\mathcal{B}(X,Y,I)$; $\mathcal{B}_T(X,Y,I)$ contains only those concepts which respect the constraints represented by AD-formulas from T, i.e. only "the interesting" concepts from $\mathcal{B}(X,Y,I)$. Note that the concept of AD-formulas is closely related to the concept of a surmise relation, see [3], and that AD-formulas are particular cases of clauses studied in [5]. In this paper, we concentrate on AD-formulas from the logical point of view. In particular, we present basic relationships between AD-formulas and attribute implications, results on models, semantic entailment, and non-redundant bases including an algorithm for their computation.

In the rest of this section, we present preliminaries. An attribute implication [4] (over Y) is an expression $A \Rightarrow B$, where $A,B \subseteq Y$. $A \Rightarrow B$ *is true in* $M \subseteq Y$, written $M \models A \Rightarrow B$, if we have: if $A \subseteq M$ then $B \subseteq M$. A set $M \subseteq Y$ is called a *model of* T if, for each $A \Rightarrow B \in T$, $M \models A \Rightarrow B$. Let $\mathrm{Mod}(T)$ denote the set of all models of T. $A \Rightarrow B$ *semantically follows from* T (T *semantically entails* $A \Rightarrow B$) if, for each $M \in \mathrm{Mod}(T)$, $M \models A \Rightarrow B$. Basic results concerning attribute implications can be found in [2, 4]. Attribute implications are also used in database theory (as functional dependencies), see [8].

An *attribute-dependency formula* [1] (shortly, an AD-formula) over Y is an expression $A \sqsubseteq B$, where $A,B \subseteq Y$. $A \sqsubseteq B$ *is true in* $M \subseteq Y$, written $M \models A \sqsubseteq B$, if we have: if $A \cap M \neq \emptyset$ then $B \cap M \neq \emptyset$. Analogously as in case of attribute implications, we introduce models and semantic entailment. A set $M \subseteq Y$ is called a *model of* T if, for each $A \sqsubseteq B \in T$, $M \models A \sqsubseteq B$. Let $\mathrm{Mod}(T)$ denote the set of all models of T. $A \sqsubseteq B$ *semantically follows from* T (T *semantically entails* $A \sqsubseteq B$) if, for each $M \in \mathrm{Mod}(T)$, $M \models A \sqsubseteq B$. For brevity, we write $y \sqsubseteq B$ ($A \sqsubseteq y$) instead of $\{y\} \sqsubseteq B$ ($A \sqsubseteq \{y\}$).

The following assertion shows a connection between validity (truth) of attribute implications and AD-formulas which will be used in the sequel.

Lemma 1 (see [1]) *For $A,B,M \subseteq Y$, we have $M \models A \sqsubseteq B$ iff $\overline{M} \models B \Rightarrow A$, where $\overline{M} = Y - M$.* □

$\mathcal{S} \subseteq 2^Y$ is a *closure* (*interior*) *system* on Y iff $\mathcal{S}$ is closed under arbitrary intersections (unions). Operator $op\colon 2^Y \to 2^Y$ is a *closure* (*interior*) *operator* if, for each $A, B \subseteq Y$, (i) $A \subseteq op(A)$ ($op(A) \subseteq A$); (ii) $A \subseteq B$ implies $op(A) \subseteq op(B)$; (iii) $op(op(A)) = op(A)$.

2. Entailment of AD-formulas

In this section we are interested in semantic entailment of AD-formulas. We show that the system of all models of a set of AD-formulas is an interior system and describe the associated interior operator. Furthermore, we show that semantic entailment of an AD-formula form a set of AD-formulas can be decided using a single model. Finally, we show that each interior operator can be described by a set of AD-formulas so that the fixed points of the interior operator coincide with models of the AD-formulas.

In [1], AD-formulas were introduced as expressions of the form $y \sqsubseteq y_1 \sqcup \cdots \sqcup y_n$, i.e., under our notation, as expressions of the form $y \sqsubseteq B$, where $B \subseteq Y$. The following assertion shows that our definition presented in Section 1. which allows for more than one attribute on the left-hand side of $\sqsubseteq$ is, in fact, an inessential extension of that one presented in [1].

Theorem 2 (i) *$M \models \{y_1, \ldots, y_m\} \sqsubseteq B$ iff, for each $i = 1, \ldots, m$, we have $M \models y_i \sqsubseteq B$.*

(ii) *For each set T of AD-formulas and each AD-formula φ, $T \models \varphi$ iff $\lfloor T \rfloor \models \varphi$, where $\lfloor T \rfloor = \{y \sqsubseteq B \,|\, A \sqsubseteq B \in T \text{ and } y \in A\}$.*

Proof. "(i)": Let $M \models \{y_1, \ldots, y_m\} \sqsubseteq B$. Thus, for each $i = 1, \ldots, m$, if $y_i \in M$ then $M \models \{y_1, \ldots, y_m\} \sqsubseteq B$ gives $M \cap B \neq \emptyset$, i.e. $M \models y_i \sqsubseteq B$.

Conversely, let $M \models y_i \sqsubseteq B$ be true for each $i = 1, \ldots, m$. Then, from $M \cap \{y_1, \ldots, y_m\} \neq \emptyset$ it follows that there is i_0 such that $y_{i_0} \in M$. Since $M \models y_{i_0} \sqsubseteq B$, we get $M \cap B \neq \emptyset$, which yields $M \models \{y_1, \ldots, y_m\} \sqsubseteq B$.

"(ii)": It suffices to check that $\mathrm{Mod}(T) = \mathrm{Mod}(\lfloor T \rfloor)$. This claim follows directly from (i). □

Theorem 2 (i) allows us to merge AD-formulas with the same right-hand side of $\sqsubseteq$ into a single AD-formula, which is true in a model iff all the original AD-formulas are true in that model. Due to Theorem 2 (ii), we are able to reduce our considerations about semantic entailment only to AD-formulas $y \sqsubseteq B$ when desirable.

Recall that the system of all models of a set of attribute implications is a closure system [4]. This pertains to sets of AD-formulas of the form $y_1 \sqsubseteq y_2$ because we have $M \models y_1 \sqsubseteq y_2$ iff $M \models \{y_1\} \Rightarrow \{y_2\}$. For general sets of AD-formulas, the models do not form a closure system as demonstrated by the following example.

Example 3 Take a set $T = \{y \sqsubseteq \{y_1, y_2\}\}$. Obviously, both $M_1 = \{y, y_1\}$ and $M_2 = \{y, y_2\}$ are models of T. On the other hand, $M_1 \cap M_2 = \{y\}$ is not a model of T.

For models of general sets of AD-formulas, we have

Theorem 4 *Let T be a set of AD-formulas. Then* $\mathrm{Mod}(T)$ *is an interior system.*

Proof. Take $\{M_i \in \mathrm{Mod}(T) \,|\, i \in I\}$. We show that $\bigcup_{i \in I} M_i$ is a model of T. Take any $A \sqsubseteq B \in T$. If $A \cap \bigcup_{i \in I} M_i \neq \emptyset$, then there is $i_0 \in I$ such that $A \cap M_{i_0} \neq \emptyset$. Since M_{i_0} is a model of T, $M_{i_0} \models A \sqsubseteq B$ gives $B \cap M_{i_0} \neq \emptyset$. Hence, $B \cap \bigcup_{i \in I} M_i \neq \emptyset$ which shows that $\bigcup_{i \in I} M_i \models A \sqsubseteq B$. Since $A \sqsubseteq B \in T$ was taken arbitrarily, we get that $\bigcup_{i \in I} M_i$ is a model of T. □

Using Theorem 4, for each set T one can consider an associated interior operator $I_T : 2^Y \to 2^Y$ defined by $I_T(M) = \bigcup\{N \in \mathrm{Mod}(T) \,|\, N \subseteq M\}$. Clearly, we have

$$\mathrm{Mod}(T) = \{M \subseteq Y \,|\, M = I_T(M)\}.$$

That is, I_T is an interior operator which assign to each $M \subseteq Y$ the greatest model $I_T(M)$ of T which is less than or equal to M. The following algorithm shows a way to compute $I_T(M)$ given T and M.

Algorithm 1
Input: set T of AD-formulas, $M \subseteq Y$
Output: $I_T(M)$

set N to M
while there is $y \sqsubseteq B \in \lfloor T \rfloor$ such that $N \not\models y \sqsubseteq B$:
  choose $y \sqsubseteq B \in \lfloor T \rfloor$ such that $y \in N$ and $B \cap N = \emptyset$
  remove y from N
return N

Proof. In order to check soundness of the algorithm, we prove that (i) N, which is returned, is a model of T such that $N \subseteq M$; and (ii) for each model $M' \in \mathrm{Mod}(T)$ such that $M' \subseteq M$ we have $M' \subseteq N$. From (i) and (ii) we immediately get that $N = I_T(M)$.

Claim (i) is easy to see: using Theorem 2 (ii), if there is no $y \sqsubseteq B \in \lfloor T \rfloor$ such that $M \not\models y \sqsubseteq B$ then M is a model of $\lfloor T \rfloor$, i.e. it is a model of T. In addition to that, $N \subseteq M$ because N is constructed by removing attributes from M.

To see (ii), first observe that N results from M by sequential removing of attributes. Hence, we can consider a sequence $M = N_0 \supset N_1 \supset \cdots \supset N_k = N$ which represents the changes of N during the computation.

We now prove the following property of each N_i ($i = 0, \ldots, k$): for each model $M' \in \mathrm{Mod}(T)$ such that $M' \subseteq M$ we have $M' \subseteq N_i$. For $N_0 = M$, the claim is trivial. By induction, assume that the claim is true for each N_j with $j < i \leq k$. N_i results by removing of an attribute from N_{i-1}. Thus, for N_i and N_{i-1} there is $y \sqsubseteq B \in \lfloor T \rfloor$ such that $N_i \cup \{y\} = N_{i-1}$ and $N_{i-1} \not\models y \sqsubseteq B$. That is, $N_{i-1} \cap B = \emptyset$. Using the assumption, we get that for each $M' \in \mathrm{Mod}(T)$

with $M' \subseteq M$, $M' \subseteq N_{i-1}$, i.e. $M' \cap B = \emptyset$. Thus, the attribute y cannot be contained in any $M' \in \mathrm{Mod}(T)$ such that $M' \subseteq M$. This yields that $N_i = N_{i-1} - \{y\}$ is greater than or equal to each model $M' \in \mathrm{Mod}(T)$ such that $M' \subseteq M$. Altogether, each N_i ($i = 0, \ldots, k$) has the desired property. Since $N = N_k$, the property yields that for each model $M' \in \mathrm{Mod}(T)$ with $M' \subseteq M$ we have $M' \subseteq N$. □

Analogously, as in case of attribute implications [4], semantic entailment of $A \sqsubseteq B$ from a set T of AD-formulas can be characterized by a single model of T as it is shown by the following theorem. Note that an analogous property is also known from logic programming where the role of "important models" is played by least Herbrand models, see [7].

Theorem 5 *Let T be a set of AD-formulas, $A \sqsubseteq B$ be an AD-formula. Then the following are equivalent.*

(i) $T \models A \sqsubseteq B$,
(ii) $I_T(\overline{B}) \models A \sqsubseteq B$,
(iii) $A \cap I_T(\overline{B}) = \emptyset$.

Proof. "(i) ⇒ (ii)": $T \models A \sqsubseteq B$ means that $A \sqsubseteq B$ is true in each model of T. Since $I_T(\overline{B})$ is a model of T, we get (ii).

"(ii) ⇒ (iii)": Let (ii) be true. We have $I_T(\overline{B}) \models A \sqsubseteq B$ iff if $A \cap I_T(\overline{B}) \neq \emptyset$ then $B \cap I_T(\overline{B}) \neq \emptyset$. The latter is true iff $B \cap I_T(\overline{B}) = \emptyset$ implies $A \cap I_T(\overline{B}) = \emptyset$. Now, since $I_T(\overline{B}) \subseteq \overline{B}$, we get $B \cap I_T(\overline{B}) = \emptyset$, yielding $A \cap I_T(\overline{B}) = \emptyset$.

"(iii) ⇒ (i)": Let $A \cap I_T(\overline{B}) = \emptyset$. It suffices to check that, for each model $M \in \mathrm{Mod}(T)$, $M \models A \sqsubseteq B$. We show that under the assumption of $B \cap M = \emptyset$ we get $A \cap M = \emptyset$. Thus, let $B \cap M = \emptyset$. This yields $M \subseteq \overline{B}$. Moreover, we get $M = I_T(M) \subseteq I_T(\overline{B})$ because I_T is monotone, and M is closed under I_T (recall that M is a model of T). Now, since we assume $A \cap I_T(\overline{B}) = \emptyset$, the latter observation $M \subseteq I_T(\overline{B})$ yields $A \cap M = \emptyset$, which proves the claim. □

Thus, Algorithm 1 together with Theorem 5 allow us to check if $T \models A \sqsubseteq B$ without the need to go through all the models of T:

Algorithm 2
Input: set T of AD-formulas and AD-formula $A \sqsubseteq B$
Output: *true* if $T \models A \sqsubseteq B$, *false* otherwise.

compute $I_T(\overline{B})$ using Algorithm 1
if $A \cap I_T(\overline{B}) = \emptyset$:
  return true
else:
  return false

Proof. Application of Algorithm 1 and Theorem 5. □

Theorem 4 says that, given T, models of T can be seen as the fixed points of interior operator I_T. We now show that for each interior operator $I : 2^Y \to 2^Y$ there is a set of AD-formulas such that the models of that set are exactly the fixed points of I. In other words, this means that each interior operator I on Y can be described by a set of AD-formulas. Taking into account Theorem 4, we get that there is a one-to-one correspondence between systems of models of AD-formulas and interior operators on Y.

The following assertion is proved indirectly. That is, instead of finding a set of AD-formulas directly for a given I, we use the facts that I naturally induces a closure operator and that such a closure operator can be fully described by a set of attribute implications. The desired result then follows using Lemma 1.

Theorem 6 *Let $I : 2^Y \to 2^Y$ be an interior operator. Then there is a set T of AD-formulas such that $I = I_T$.*

Proof. Take an interior operator $I : 2^Y \to 2^Y$. Consider an operator $C : 2^Y \to 2^Y$ defined by $C(A) = \overline{I(\overline{A})}$. It is a well-known fact that such a C is a closure operator. In addition to that, for the closure operator C there is a set T' of attribute implications such that

$$M = C(M) \quad \text{iff} \quad M \in \mathrm{Mod}(T'). \qquad (*)$$

One can take, for instance, $T' = \{A \Rightarrow C(A) \,|\, A \subseteq Y\}$. Put $T = \{B \sqsubseteq A \,|\, A \Rightarrow B \in T'\}$. Using Lemma 1, we have

$$M \in \mathrm{Mod}(T') \quad \text{iff} \quad \overline{M} \in \mathrm{Mod}(T). \qquad (**)$$

Obviously, we have

$$M = C(M) = \overline{I(\overline{M})} \quad \text{iff} \quad \overline{M} = I(\overline{M}).$$

Putting $(*)$–$(***)$ together, we get that

$$\overline{M} = I(\overline{M}) \quad \text{iff} \quad \overline{M} \in \mathrm{Mod}(T) \qquad (***)$$

is true for each $M \subseteq Y$. Hence, models of T are exactly the fixed points of I, i.e. $I = I_T$. □

3. Non-redundant bases

In this section we are interested in non-redundant bases of sets of AD-formulas. In the standard database terminology, non-redundant bases are also known as non-redundant covers [8]. Described verbally, a non-redundant basis of a set T of AD-formulas is a set T' of AD-formulas which entails exactly the same AD-formulas as T, and T' is a least set with this property. Thus, non-redundant bases describe the same information (via semantic entailment) as the original sets of AD-formulas and are, in a sense, the least sets of AD-formulas that do the job. This has particular relevance in data mining (smaller set of formulas is sufficient to describe a larger one) and computation efficiency (e.g., Algorithm 2 runs faster with smaller T's).

We first introduce the notion of an equivalence of sets of AD-formulas and show its basic properties. Given sets T_1, T_2 of AD-formulas, we say that T_1 *and* T_2 *are equivalent*, written $T_1 \equiv T_2$, if for each $\varphi \in T_1$ and $\psi \in T_2$ we have $T_1 \models \psi$ and $T_2 \models \varphi$.

Theorem 7 *Let T_1, T_2 be sets of AD-formulas. Then the following are equivalent.*

(i) $\mathrm{Mod}(T_1) = \mathrm{Mod}(T_2)$,
(ii) *for each AD-formula* φ, $T_1 \models \varphi$ *iff* $T_2 \models \varphi$,
(iii) $T_1 \equiv T_2$.

Proof. "(i) $\Rightarrow$ (ii)": Follows by definition of "$\models$".

"(ii) $\Rightarrow$ (iii)": Take $\varphi \in T_1$. We have, $T_1 \models \varphi$ because φ is true in each model of T_1. Now, (ii) gives $T_2 \models \varphi$. Dually, $\psi \in T_2$ gives $T_1 \models \psi$. Hence, $T_1 \equiv T_2$.

"(iii) $\Rightarrow$ (i)": Let $T_1 \equiv T_2$. By contradiction, assume $\mathrm{Mod}(T_1) \neq \mathrm{Mod}(T_2)$. Thus, either $\mathrm{Mod}(T_1) \not\subseteq \mathrm{Mod}(T_2)$ or $\mathrm{Mod}(T_2) \not\subseteq \mathrm{Mod}(T_1)$. Suppose $\mathrm{Mod}(T_1) \not\subseteq \mathrm{Mod}(T_2)$. Then, there is $M \in \mathrm{Mod}(T_1)$ such that $M \notin \mathrm{Mod}(T_2)$. This means that there is $A \sqsubseteq B \in T_2$ such that $M \not\models A \sqsubseteq B$. On the other hand, $T_1 \equiv T_2$ yields $T_1 \models A \sqsubseteq B$, i.e. $M \models A \sqsubseteq B$ because M is a model of T_1, a contradiction. □

Coming back to our initial motivation, for a given T we wish to find a least T' such that $T \equiv T'$. More precisely, a set T' of AD-formulas is called a *non-redundant basis of* T if $T \equiv T'$ and there is no $T'' \subset T'$ with $T'' \equiv T$. A set T' of AD-formulas is called a *minimal basis of* T if $T \equiv T'$ and for each T'' such that $T \equiv T''$, we have $|T'| \leq |T''|$. Obviously, if T' is a minimal basis of T, then T' is a non-redundant basis of T (but not *vice versa* in general). Hence, we are interested in generating of non-redundant (or minimal) bases of given sets of AD-formulas.

A naive way to find a non-redundant basis of T is to start with T and remove AD-formulas from T until no AD-formula from T follows from the others. This procedure is, however, inefficient, and does not ensure that a non-redundant basis found this way is minimal.

In the sequel we present an algorithm which generates a minimal basis of any T. Again, we use an indirect procedure together with observations from Theorem 6. Namely, we first associate with any T a closure operator C, then use Ganter's NEXTCLOSURE algorithm [4] to compute a minimal basis (of attribute implications) related to C, and then use the set to determine a minimal basis of T. Due to the lack of space, we omit detailed description of the algorithm which will be available in full version of this paper. Let us mention that the algorithm allows us to list a minimal basis of T with a polynomial time delay, cf. [4, 6]. The algorithm is sketched below.

Algorithm 3
Input: set T of AD-formulas
Output: minimal basis $T_{\min}$ of T

define $C(M)$ *by* $\overline{I_T(\overline{M})}$
if $C(\emptyset) \neq \emptyset$:
  set T' *to* $\{\emptyset \Rightarrow C(\emptyset)\}$
else:
  set T' *to* $\emptyset$
set P *to* $\emptyset$
while $P \neq Y$:
  set P *to* NEXTCLOSURE (P, T')
  if $P \neq C(P)$:
    add $P \Rightarrow C(P)$ *to* T'
set $T_{\min} = \{B - A \sqsubseteq A \,|\, A \Rightarrow B \in T'\}$
return $T_{\min}$

Proof. Omitted due to the limited scope of this paper. □

Example 8 Take the following set of AD-formulas:

$$T = \{a \sqsubseteq \{b,c,e\}, b \sqsubseteq \{c,d,e\}, b \sqsubseteq \{b,c,d\}, \\ c \sqsubseteq \{a,b,d\}, d \sqsubseteq \{c,e\}, d \sqsubseteq \{b,d\}, e \sqsubseteq \{c,d\}\}.$$

The minimal basis $T_{\min}$ of T which is computed by Algorithm 3 is the following:

$$\{\{c,e\} \sqsubseteq \{a,b,d\}, \{a,b,e\} \sqsubseteq \{c,d\}, \{a,b,d\} \sqsubseteq \{c,e\}\}.$$

Hence, compared to the original set T which consists of seven AD-formulas, $T_{\min}$ consists of three AD-formulas which encompass the same information about attribute dependencies as T.

Acknowledgement Supported by grant No. 1ET101370417 of GA AV ČR, by grant No. 201/05/0079 of the Czech Science Foundation, and by institutional support, research plan MSM 6198959214.

References

[1] Bělohlávek R., Sklenář V.: Formal concept analysis constrained by attribute-dependency formulas. In: Ganter B., Godin R. (Eds.): ICFCA 2005, *LNCS* **3403**, pp. 176–191, Springer-Verlag, Berlin/Heidelberg, 2005.

[2] Carpineto C., Romano G.: *Concept Data Analysis. Theory and Applications.* J. Wiley, 2004.

[3] Diognon J.-P., Falmagne J.-C.: *Knowledge Spaces*. Springer, Berlin, 1999.

[4] Ganter B., Wille R.: *Formal Concept Analysis. Mathematical Foundations.* Springer, Berlin, 1999.

[5] Ganter, Wille: Contextual attribute logic. In: Tepfenhart W., Cyre W. (Eds.): *Proceedings of ICCS 2001*, Springer, 2001.

[6] Guigues J. L., Duquenne V.: Familles minimales d'implications informatives resultant d'un tableau de données binaires. *Math. Sci. Humaines* **95**(1986), 5–18.

[7] Lloyd, J. W.: *Foundations of Logic Programming* (2nd ed.). Springer-Verlag, New York, 1987.

[8] Maier D.: *The Theory of Relational Databases.* Computer Science Press, Rockville, 1983.