FAST ALGORITHM FOR CONSTRUCTION OF ALL INTERSECTIONS OF OBJECTS FROM A FINITE SEMILATTICE

S. O. Kuznetsov

Nauchno-Tekhnicheskaya Informatsiya, Seriya 2,
Vol. 27, No. 1, pp. 17-20, 1993

UDC 519.688

We propose an algorithm constructing all elements of a semilattice from a set of generating elements G. The problem of computation of the number of all elements is $\#P$-complete even in case of a Boolean semilattice, i.e., when G is a family of subsets of a certain set U. The number of intersections can be exponential with respect to the size of the set of generating elements. This algorithm is optimal because it generates all intersections within a time which is linear with respect to their number.

1. INTRODUCTION: BASIC DEFINITIONS

Information problems encountered in data processing often require finding similarities in descriptions of objects. When objects are represented by sets, similarities can be expressed in set-theoretic intersections of descriptions. The similarity for data of a more complex type can be defined by a semilattice operation, which has the following properties for arbitrary objects X, Y, Z in its action domain:

$X \sqcap X = X$ (idempotence),

$X \sqcap Y = Y \sqcap X$ (commutativity),

$(X \sqcap Y) \sqcap Z = X \sqcap (Y \sqcap Z)$ (associativity).

$X \sqcap 0 = 0$ for a certain element 0 of the semilattice, which is referred to as the zero element.

Operation $\sqcap$ defines a natural order $\sqsubseteq$: $X \sqsubseteq Y \Leftrightarrow X \sqcap Y = X$.

The similarity search in certain information systems is reduced to taking pairwise or n-ary intersections for a fixed n [1]. However, in certain situations [2-6] one needs to find all intersections formed by an arbitrary number of objects from a database. Algorithms executing this task posses an exponential complexity, even when the data are represented by sets, because of the exponential number of potential intersections [7]. Estimating the number of various intersections is $\#P$-complete problem [8]. This makes it necessary to develop algorithms which solve this problem in a most effective manner. We propose an algorithm which finds intersections for an arbitrary semilattice. The estimate of its operation speed proves that it is optimal in a certain sense.

We make use of notations from [3] in definitions of the basic concepts.

Suppose S is a finite set of objects, for which an idempotence commutativity and associativity operation $\sqcap$ is defined.

Definition 1.1. Suppose $X_1, ..., X_k \in S$, in that case $(\{X_1, \ldots, X_k\})' = X_1 \sqcap \ldots \sqcap X_k$. If $h \sqsubseteq X_i$ for a certain $X_i \in S$, then by definition $h' = \{X \mid X \in S,\ X \sqsupseteq h\}$. We suppose also $(\{X_1, \ldots, X_k\})'' = \{X \in S \mid X \sqsupseteq X_1 \sqcap \ldots \sqcap X_k\}$. Set $(\{X_1, ..., X_k\})''$ is referred to as the closure of set $\{X_1, ..., X_k\}$ (it can readily be verified that operation $''$ indeed has the property of a closure operator: extensivity, idempotence, and isotonicity).

If set Y is closed, i.e., $(Y)'' = Y$, then the set of objects $(Y \cup \{X_i\})'' = Y \cup \{X_i\} \cup \{X_{i_1}, ..., X_{i_m}\}$ will be denoted by $[(Y, X)X_{i_1}, ..., X_{i_m}]$.

Definition 1.2. We say that a pair of closed sets $\langle h, \{X_1, ..., X_n\}\rangle$ is an exhaustive intersection (EI) with respect to set S, if $\{X_1, ..., X_n\} \subseteq S$, $(\{X_1, ..., X_n\})' = h$ and $h' = \{X_1, ..., X_n\}$. h is said to be an intersection, and $\{X_1, ..., X_n\}$ is referred to as the set of parents or Par-set, denoted Par(h).

EI in [3] is called a concept, and h is called its intent, while $\mathrm{Par}(h)$ is referred to as its extent. Pairs of the form $\langle X, \{X\}\rangle, X \in S$ will be called singular EI. X will be referred to as singular intersection.

We assume that the order in semilattice of EI (and accordingly, the direction "top to bottom" or "bottom to top," a distinction which is important for the construction of an algorithm) is defined by the order at intersections, so that $\langle h_1, \mathrm{Par}(h_1)\rangle \preceq \langle h_2, \mathrm{Par}(h_2)\rangle$ if and only $h_1 \sqsubseteq h_2$ and, accordingly, $\mathrm{Par}(h_1) \subseteq \mathrm{Par}(h_2)$.

The set of all EI, i.e., the set of pairs consisting of semilattice elements and their parents generated from set S will be denoted MIP(S).

The problem solved by the algorithm constructed in the present paper thus consists in generating MIP(S) from given S.

The dimension of the input of the algorithm generating all EI is characterized by the number of initial objects, i.e., quantity $|S| = m$ and the size of maximum object N. In the special case of Boolean lattice $N = |U| = n$, where U is the carrier of the Boolean. Since the dimension of set MIP(S) can be exponential with respect to S [7], and the problem of defining this dimension is #P-complete [8], the estimate of the effectiveness should include the dimension of MIP(S), i.e., $|\mathrm{MIP}(S)| = K$. The algorithm whose time complexity is a linear function of K and the polynomial with respect to the input dimension (i.e., a polynomially enumerating algorithm according to the terminology of [9]) is said to be effective.

2. "CLOSE ONE" ALGORITHM

A version of the "close one" (CO) algorithm, which does not make use of the advantages of a lexicographic ordering, has been described in [10] (the "VP-bt tree" algorithm) for the "top to bottom" strategy.

We assume that objects of S are enumerated, and each set X of objects on S is represented by an ordered set, where elements are arranged according to their numeration on S. The order on objects of S also induces a lexicographic ordering of sets from 2^S. The generation of all intersections is a "top to bottom" process constructing a tree with EI assigned to its nodes. Objects of S can be marked or remain unmarked in the course of tree construction and traversal independently at any node. According to the following description, a tree is generated "first into depth," although alternative strategies are also conceivable.

Step 0. The root node of a tree is linked to m nodes. Each node is in a one-to-one correspondence to an object from S. All objects at the nodes are unmarked $Y := \varnothing$.

Step 1. Take first unmarked element of S. Suppose it is X_i. Compute $(Y \cup \{X_i\})'$ and $(Y \cup \{X_i\})'' = \{(Y, X_i) X_{i_1}, \ldots, X_{i_m}\}$ for certain $X_{i_1}, \ldots, X_{i_m} \in S$. Form a new node corresponding to $(Y \cup \{X_i\})''$. Link this node by an edge with the node corresponding to Y.

Step 2. If for a certain i_j we have $i_j \le i$, then EI $(Y \cup \{X_i\})''$ has already been generated previously (see Lemma 2.3). We mark all elements of S at node $(Y \cup \{X_i\})''$. This tree branch will thus not be continued. If the EI has not been generated before, we mark additionally at node Y element X_i and at node $(Y \cup \{X_i\})''$ all elements of set $(Y \cup \{X_i\})''$.

Step 3. If all elements of S have been marked at $(Y \cup \{X_i\})''$, go to Step 4. Otherwise $Y := (Y \cup \{X_i\})''$, go to Step 1.

Step 4. Return up the tree to the nearest node which has unmarked elements of S. If such a node exists and corresponds to set of objects Z, then $Y := Z$, go to Step 1. If there is no such node, or intersections have been constructed, the algorithm stops.

We will give an example where CO algorithm generates certain intersections and exponential number of times.

Consider a "top to bottom" strategy for a Boolean semilattice on carrier $U = \{a_1, \ldots, a_n\}$ and generating set $S = \{X_1, \ldots, X_m, X_{n+1}\}$, where $X_i = U \setminus \{a_i\}, 1 \le i \le n, X_{n+1} = \{a_n\}$. We recall that the "top to bottom" or "bottom to top" direction is determined by the order on EI: in the former case, the process begins from generating EI $\langle h, \mathrm{Par}(h)\rangle$ with the embedding-largest intersections, i.e., h, and passes to smaller intersections. In that case EI $\langle \{a_n\}, \{X_1, \ldots, X_{n-1}, X_{n+1}\}\rangle$ can be obtained from an arbitrary larger EI $\langle h, \mathrm{Par}(h)\rangle$ (i.e., $\mathrm{Par}(h) \subseteq \mathrm{Par}(\{a_n\})$) by closure $(h \cup \{X_{n+1}\})'$. Since there are exactly $2^n - 1$ such EI, we can obtain EI $\langle \{a_n\}, \mathrm{Par}(\{a_n\})\rangle$ as many times.

Despite an apparently pessimistic conclusion suggested by this example, the actual number of such EI and the total number of all EI (including multiple ones) generated by this algorithm is not large. This is implied by

Lemma 2.1. The number of exhaustive intersections produced by CO algorithm in the "top to bottom" strategy is $O(mK)$.

Proof. CO algorithm continues only one branch out of all tree branches leading to a given EI (see step 2). Therefore, for each EI $\langle h, \mathrm{Par}(h)\rangle$ the number of EI obtained by the CO algorithm directly after $\langle h, \mathrm{Par}(h)\rangle$ may not be larger than the number of objects belonging to $S \backslash \mathrm{Par}(h)$, i.e., not greater than $m-k < m$. Since there are K of EI, the total number of intersections produced by the CO algorithm does not exceed mk□.

Lemma 2.2. Suppose $S = \{X_1, ..., X_m\} \subseteq 2^U, U = \{a_1, ..., a_n\}$. The number of applications of the intersection operation, by the CO algorithm in the "top to bottom" strategy then does not exceed nK.

Proof. The set of elements of semilattice MIP(S) (denoted $\langle S, \cap, 0\rangle$), which is formed by exhaustive intersections of sets from family S can be obtained by the "bottom to top" strategy as the set of elements of semilattice $\langle S', \cup, 0\rangle$, i.e., MIP($S$) with the set of parents S' that are dual with respect to S: $S' = \{Y_i | Y_i = \{X_j | a_i \in X_j, 1 \le i \le n, 1 \le j \le m\}\}$ and semilattice operation $\cup$, which is the Boolean union operation. Since $|S'| = n$, therefore, by virtue of Lemma 2.1, the number of applications of the intersection operation by CO algorithm does not exceed nk□.

It follows from Lemmas 2.1 and 2.2 that the memory capacity of the CO algorithm is a linear function of the semilattice elements, i.e., K. The estimate of the operation speed of this algorithm consists of the EI generation time and the time of testing the fact that a given EI has not been generated before. A direct check with the scan of entire EI list takes $O(K^2)$ operations (in this case, matched test). It can be shown, however, that the uniqueness of generated EI can be tested within a time linear with respect to K.

Suppose an arbitrary EI $H = \langle h, \mathrm{Par}(h)\rangle$ has been produced by a CO algorithm operating "top to bottom", i.e., closing one element at a time from S (for the Boolean case in the "bottom to top" strategy, i.e., closure of one element of U at a time, the following arguments apply by analogy).

A derivation of EI $\langle h, \mathrm{Par}(h)\rangle$ will be defined as a path in CO tree which leads to a certain node corresponding to $\langle h, \mathrm{Par}(h)\rangle$. A path represents a sequence of elements through which closure at the nodes of the path occurs.

An order on S elements induces a lexicographic ordering of derivations: derivation $W_1 = \langle w_1, ..., w_k\rangle$ for EI $\langle h, \mathrm{Par}(h)\rangle$ lexicographically precedes derivation $W_2 = \langle v_1, ..., v_s\rangle$, if for the first i, which is such that $w_i \neq v_i$, we have $w_i \prec v_i$ (in the sense of ordering on serial numbers of S elements).

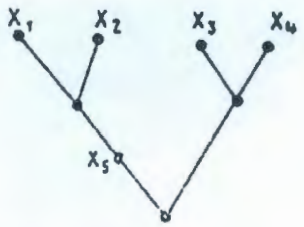
The lexicographically first derivation will be referred to as canonical derivation of EI $\langle h, \mathrm{Par}(h)\rangle$.

Consider path $X_1, ..., X_k$ in CO tree leading to EI $\langle h, \mathrm{Par}(h)\rangle$. We denote the intersections obtained by these closures, $h_1, ..., h_k$, respectively. Suppose that at the i-th step of the path, EI $\langle h, \mathrm{Par}(h)\rangle$ is generated, where

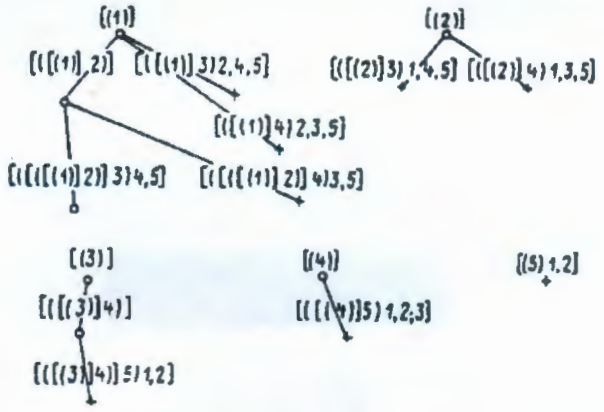
$$h_i = (\mathrm{Par}(h_{i-1}) \cup \{X_i\})' \text{ and}$$
$$\mathrm{Par}(h_i) = \mathrm{Par}(h_{i-1}) \cup \{X_{i_1}, \ldots, X_{i_s}\} \cup \{X_i\}$$

for certain $X_{i_1}, ..., X_{i_s} \in S$. We will say that X_i is a leading element and $X_{i_1}, ..., X_{i_s}$ are led elements for the i-th step of derivation $X_1, ..., X_k$. We introduce the following notation: if set Y is closed with respect to object X_i, and objects $X_{i_1}, ..., X_{i_s}$ are led elements in this case, we write the result of the closure, i.e., $(Y \cup \{X_i\})''$ as $[(Y, X_i)X_{i_1}, ..., X_{i_s}]$.

We give an example of the operation of CO algorithm for a semilattice with generating set $\{X_1, X_2, X_3, X_4, X_5\}$ and the diagram



The input data are specified by the set of objects numbered 1, 2, 3, 4, and 5. For brevity's sake, the objects in the tree are represented by their serial numbers and the root node is omitted:



The tree nodes corresponding to canonical paths are marked by circles. The remaining (non-canonical) nodes are marked by the plus sign. Note that the canonical derivation of objects from the generating set, which, as X_5, are contained in other objects of the generating set can start from nodes corresponding to different objects.

Lemma 2.3. Derivation $\langle X_1, ..., X_k \rangle$ in a CO tree is canonical if and only if for an arbitrary i all the led objects $X_{i_1}, ..., X_{i_s}$ have serial numbers greater than the numbers of leading objects $X_1, ..., X_{i-1}$ corresponding to the first $i-1$ derivation steps.

Proof. 1) Necessity. Suppose that the condition of the lemma is not satisfied and for certain i, j, t ($1 \le t \le i, 1 < j < s$) serial number X_{i_j} is less than the number of X_t in the numeration of objects from S. If $X_t \in (X_1 \cup ... \cup X_{i-1} \cup X_i)^*$, then path $\langle X_1, ..., X_{i-1}, X_{i_j}, X_{i+1}, ..., X_i \rangle$ leads to EI $\langle h_i, \mathrm{Par}(h_i) \rangle$ because the objects which are led for X_t are also led for X_{i_j}. If $X_t \notin (X_1 \cup ... \cup X_{i-1} \cup X_i)^*$, then we arrive at EI $\langle h_i, \mathrm{Par}(h_i) \rangle$ via path $\langle X_1, ..., X_{i-1}, X_{i_j}, X_t, X_{i+1}, ..., X_i \rangle$. In any case, each of these paths lexicographically precedes $\langle X_1, ..., X_t, ..., X_i \rangle$. Therefore, derivation $\langle X_1, ..., X_t, ..., X_i \rangle$ is not canonical.

2) Sufficiency follows directly from the definition of a canonical derivation. Indeed, if the condition of the lemma is satisfied, then the closure at each step occurred at the element with the least serial number □.

Corollary. Derivation $\langle X_1, ..., X_{k-1}, X_k \rangle$ in CO tree is canonical if and only if derivation $\langle X_1, ..., X_{k-1} \rangle$ is canonical and the serial number of each led object for the k-th step is greater than the serial number of object X_{k-1}.

Testing at each step in the construction of the CO tree the condition of Lemma 2.3, we can avoid the scanning of the entire list of generated EI to determine whether a given EI has or has not been constructed earlier: one needs only to check the canonicity of the derivation and then avoid continuing the paths from the nodes to which a non-canonical derivation has led. The canonicity of a derivation at each node is tested according to the corollary of Lemma 2.3 by comparing the serial number of each $X_{i_1}, ..., X_{i_j}$ with $i-1$, i.e., it is executed in $O(m)$ operations with 0-1 vectors.

The complexity of the generation of each node must be taken into account in estimating the time complexity. One $\sqcap$ operation and at most m embedding test are needed for a closure. To test the canonicity of a derivation leading to a given node, one has to perform, by virtue of Lemma 2.3, at most m^2 operations with 0-1 lines. The total time complexity of CO algorithm is $O(mK)\$ + O(m^2K)\% + O(m^2K)@$, where \$ is the execution time of operation $\sqsubseteq$, % sign is the embedding testing time (more exactly, testing the executability of partial order relation $\sqsubseteq$ corresponding to semilattice operation $\sqcap$ on a pair of objects from semilattice), and the @ sign is the time of execution of operations with 0-1 lines. Specifically, in the Boolean case, the total time complexity is $O(m^2K)@$. In the "bottom to top" strategy, one has to perform m embedding tests and m intersections for the closure, and n operations to check canonicity of derivation, which amounts to $O((m+n)nK)@$ time.

In case of a greater intersection complexity, actual programs based on CO algorithm can, when continuing a tree, cross current EI with certain EI formed from examples, rather than examples as such, which have in the CO tree a common pattern with the given EI (due to the associativity of the similarity operation): an intersection with subobjects, i.e., with lesser objects requires less time than intersection with initial objects, as is the case, for instance, with graph sets.

Lemma 2.4. For an arbitrary input $\langle S, \sqcap, 0 \rangle$ CO tree is complete, i.e., each EI is generated by this algorithm at least once.

The proof of this fact is reduced to construction of a canonical derivation for an arbitrary EI $\langle h, \mathrm{Par}(h) \rangle$. For the "top to bottom" strategy one takes from set $\mathrm{Par}(h)$ the lexicographically first element X and then executes

the sequence of actions described in CO algorithm, replacing set S by set Par(h). The closure of the join of the set of examples obtained at preceding set and example from Par(h) which has the least serial number among remaining ones is thus executed at each step. For each EI from the resulting sequence the set of examples, which is contained in Par(h), includes a proper subset of the set of examples for preceding EI. Since $(\text{Par}(h))'' = \text{Par}(h)$, the system generates $\langle h, \text{Par}(h)\rangle$ at a certain step in that sequence. For the "bottom to top" strategy, similar arguments apply in case of a Boolean lattice.

All the results in this section, are summarized in the following theorem.

Theorem 2.5. The set of all elements of an arbitrary semilattice $\langle S, \sqcap, 0\rangle$, where S is the set of elements generating the semilattice, can be constructed by using memory capacity $O(mNK)$ and time $O(mK)\$ + O(m^2K)\% + O(m^2K)@$, where K is the number of elements in the semilattice, N is the size of the largest element of the semilattice, \$ is the execution time of operation $\sqcap$, % is the checking time for executability of partial order relation corresponding to semilattice operation $\sqcap$ on a pair of objects from the semilattice, and @ is the time of execution of operations with 0-1 lines. For a Boolean lattice of size n, the generation is possible with $O(mnK)$ memory capacity and $O((m + n)nK)$ operations on Boolean vectors.

3. OTHER ALGORITHMS

MP algorithm is obtained from CO algorithm by modifying the first step. The MP algorithm selects additionally at each node those X_i for which values of $(Y \cup \{X_i\})''$ are embedding-maximum among all $(Y \cup \{X_j\})''$ from previously unmarked X_j. The tree is continued only along branches corresponding to these minimal closures. The MP algorithm was first formulated by Anshakov and Khazanovskii [10] in a dual variant, i.e., for the "bottom to top" strategy in the Boolean case (the minimal intersection algorithm). An analog of Lemma 2.2 holds for this algorithm and so does the assertion of the linearity of memory capacity required for its operation. The memory capacity for the MP algorithm is smaller than that required by the CO algorithm (while retaining the order of magnitude $O(mK)$). This is because, from each EI, the program passes only to embedding-maximum smaller EI. However, additional operations are needed to check embeddability. Furthermore, there is no canonical derivation for MP in the general case (because the embedding-maximal EI may not include EI with lexicographically first derivations). To check the uniqueness of an EI, one has to compare it with all previously generated EI, which produces a time complexity of order $O(K^4)$. An algorithm with linear memory and size that generates all intersections may be based on checking coincident intersections using two or three balanced trees rather than testing the canonicity of derivations [11]. However, this technique is inconvenient for a parallel architecture, because it presumes coordination of processor accesses to a centralized intersection storage.

* * *

The author thanks D. P. Skvortsov and O. M. Anshakov for pointing out some inaccuracies in the initial draft of this paper.

REFERENCES

1. A. B. Rozenblit and V. E. Golender, Logical/Combinatoic Methods in Design of Pharmaceutical Drugs [in Russian], Zinatne, Riga, 1983.

2. V. K. Finn, "Plausible inference in JSM intelligent systems," Itogi Nauki i Tekhniki, Ser. Informatika, vol. 15, pp. 54-101, 1991.

3. R. Wille, "Restructuring lattice theory: an approach based on hierarchies of concepts," in: Ordered Sets, ed. I. Rival, Reidel, Dordrecht, pp. 445-470, 1982.

4. V. P. Gladun and N. D. Vashchenko, "On walkthrough problems in automatic hypothesis generation by the JSM method," Kibernetika, no. 2, pp. 107-112, 1975.

5. J. Duda, "Boolean concept lattices and good concepts," Cas. Pest. Mat., no. 114, pp. 165-175, 1989.

6. M. Novotny and Z. Pawlak, "Algebraic theory of independence in information systems," Fundamenta Informaticae, no. 14, pp. 454-476, 1991.

7. M. I. Zabezhailo, "Some scanning problems in automatic hypothesis generation by the JSM-method," Nauchno-Tekhnicheskaya Informatsiya, Ser. 2, no. 1, pp. 28-31, 1988.

8. S. O. Kuznetsov, "Graph interpretation and complexity of problems of the search for specific types of regularities," Nauchno-Tekhnicheskaya Informatsiya, Ser. 2, no. 1, pp. 23-28, 1989.

9. L. G. Valiant, "The complexity of enumeration and reliability problems," SIAM J. Comput., vol. 8, no. 1, pp. 410-421, 1979.

10. M. I. Zabezhailo, V. G. Ivashko, S. O. Kuznetsov, M. A. Mikheenkova, K. P. Khazanovskii, and O. M. Anshakov, "Algorithmic and programming aspects of the JSM method of automatic hypothesis generation," Nauchno-Tekhnicheskaya Informatsiya, Ser. 2, no. 10, pp. 1-14, 1987.

11. A. Ako, J. Hopcroft, and J. Ulman, Construction and Analysis of Computational Algorithms [Russian translation], Mir, Moscow, 1979.

4 December 1993