# RECOVERY OF INCOMPLETE TABLES UNDER FUNCTIONAL DEPENDENCIES

Miyano, Satoru
Department of Mathematics, Kyushu University

Haraguchi, Makoto
Research Institute of Fundamental Information Science, Kyushu University

https://doi.org/10.5109/13334

*Bulletin of Informatics and Cybernetics 20 (1982)*

# RECOVERY OF INCOMPLETE TABLES UNDER FUNCTIONAL DEPENDENCIES

By

**Satoru MIYANO***

**and**

**Makoto HARAGUCHI****

(Received November 2, 1981)

## ABSTRACT

In this paper we consider incomplete tables over some attribute set and discuss the recovery problem under functional dependencies. An incomplete table is introduced as a table over an attribute set such that each table entry is a subset of the attribute domain instead of just a single value. The recovery problem we consider is to extend the given incomplete table so that each table entry contains only one value and the resulting table will be consistent with a given set of functional dependencies. We show that the recovery problem for incomplete tables such that each table entry is finite is **NP**-complete. We also give some observation on the unique recoverability. Furthermore, we consider time-variant tables. The recovery problem for time-variaut tables is shown to be **PSPACE**-complete.

## 0. INTRODUCTION

Much has been written about management of data with complete information. However, there is a view that information we have in the real world is sometimes incomplete. By information incompleteness, we mean that we have a collection of values in which the actual value is believed to be in, but the actual value is unknown.

Recently several approaches have been made to give semantics for a query language for a database with incomplete information and logical problems about incomplete information databases have been studied in the original work by Lipski (1979, 1981). The notion of functional dependency is extended for relational databases with null values and a complete axiomatization of inference rules for extended functional dependencies is given in Vassiliou (1980). A logic concerned with incomplete information is also developed in Ono and Nakamura (1980) and Nakamura (1981) and a mathematical model of an information storage and retrieval system with incomplete information is given in Jaegermann (1978, 1979).

---

* Department of Mathematics, Kyushu University 33, Fukuoka 812, Japan.

** Research Institute of Fundamental Information Science, Kyushu University 33, Fukuoka 812, Japan.

In contrast with these approaches to the incomplete information management, we consider the problems arising from incomplete information recovery and study their computational complexity. In general, the problem we are now facing with is the following interesting problem:

Given an incomplete information, for example, some part of the information is lost or has some mistakes, if we have some knowlege about the information with which we are concerned, how can we recover this incomplete information or is it at least possible to recover the incomplete information so that it will be consistent with the knowledge we have?

In this paper, we will focus our attention on the relational model of data and functional dependencies (Codd (1970, 1971)). We consider incomplete tables and discuss the complexity of the recoverability for incomplete tables of two types. One types is called the finite type meaning that each table entry is a finite set. The other type is called the infinite type meaning that each table entry is a single value or the attribute domain itself that is assumed to be infinite. We firstly show that the recoverability problem for incomplete tables of finite type is $\boldsymbol{NP}$-complete. Observing that there may be many recoveries from a given incomplete table, we consider the uniqueness of recovery. Our result states that deciding whether a given incomplete table of infinite type has the unique recovery requires only polynomial time amount. Furthermore, we show that this problem is complete for $\boldsymbol{P}$, the class of problems solvable in deterministic polynomial time, with respect to log space reductions. On the other hand, in the case of finite type incomplete tables, the unique recovery problem is proved to be co-$\boldsymbol{NP}$-hard. We also consider time-variant tables. A time-variant table is a two-way infinite sequence of tables with the same attribute set and the same number of rows. The notion of time-variant table is realistic in the sense that the tables are updated from time to time obeying some rules. The problem we consider is the following: We are assumed to have time-invariant knowledge about the relations between the tables of "yesterday" and "today" and the time-invariant knowledge on the table itself. Then given a table at one time unit whose singly valued entries represent the time-invariant information, can we recover the incomplete table to a complete table so that it will be at least consistent with the time-invariant information and the relations which hold for every two consecutive days? Our result is that the recoverability problem for time-variant incomplete tables of finite type is $\boldsymbol{PSPACE}$-complete.

## 1. HOW HARD IS THE RECOVERY PROBLEM?

Firstly, we give some basic definitions which are related to the relational model of data.

Definition 1.1. Let $U$ be a set called an *attribute set*. For each element $A$ in $U$, we associate a set $D(A)$ called the domain of the attribute values of $A$ or the attribute domain of $A$. An *incomplete table* $T$ over an attribute set $U$ is a matrix indexed by $\{1, \cdots, n\} \times U$ for some $n \geqq 0$ such that for each attribute $A$ in $U$ the entries in the column corresponding to the attribute $A$ are nonempty subsets of $D(A)$.

Example 1.2.

| NAME | AGE | SAL | SEX |
|---|---|---|---|
| {AKIRA} | [40, ∞] | (0, ∞) | {M} |
| {KAORU} | [25, 40] | (30000, ∞) | {M, F} |
| {YOKO} | [35, 45] | (0, 50000) | {F} |
| {DANA} | [6, 35] | (0, 30000) | {M, F} |

DEFINITION 1.3. An incomplete table $T$ is said to be *complete* if each table entry of $T$ is a singleton.

DEFINITION 1.4. In this paper we consider incomplete tables of the following types:

(1) Each attribute domain contains infinitely many elements and for each table entry $E$, the set $E$ is a singleton or $E$ coincides with the domain of the attribute which corresponds to $E$. Incomplete tables of this kind are said to be of *infinite type*.

(2) For each table entry $E$, the set $E$ is a finite set. Incomplete tables satisfying this condition are said to be of *finite type*.

REMARK. For the purpose of encoding, we assume each attribute domain consists of strings over some finite alphabet. For incomplete tables of infinite type, we use distinct variables $x[i]$, $i=1, \cdots$ to represent the entries which coincide with the attribute domains. We encode these incomplete tables by the conventional tuple by tuple arrangement.

EXAMPLE 1.5. (1) Let $T$ be

| AGE | SAL | SEX |
|---|---|---|
| {25, 26} | {10000, 20000} | {M} |
| {30} | {10000, 30000} | {M, F} |

Then the encoding of $T$ is

[AGE SAL SEX] ({25, 26} {10000, 20000} {M}) ({30} {10000, 30000} {M, F}).

(2) Let $T$ be

| INSTRUCTOR | ADDRESS | PHONE# | SPECIALIZATION |
|---|---|---|---|
| GAUSS | $x[1]$ | $x[2]$ | MATHEMATICS |
| EULER | $x[3]$ | $x[4]$ | MATHEMATICS |
| EINSTEIN | $x[5]$ | $x[6]$ | PHYSICS |
| $x[7]$ | OAK_TOWN | 672-1493 | MATHEMATICS |

Its encoding is

[INSTRUCTOR ADDRESS PHONE# SPECIALIZATION] (GAUSS $x[1]$ $x[2]$ MATHEMATICS) (EULER $x[3]$ $x[4]$ MATHEMATICS) (EINSTEIN $x[5]$ $x[6]$ PHYSICS) ($x[7]$ OAK_TOWN 672-1493 MATHEMATICS).

DEFINITION 1.6. Let $T$ be a complete table over an attribute set $U$. Let $X$ and $Y$ be nonempty subsets of $U$. We say that the *functional depencency* $X \rightarrow Y$ holds in $T$ if all rows in $T$ which agree on $X$ also agree on $Y$. Given a family $F$ of functional dependencies, we say that $T$ is *consistent* with $F$ if each functional dependency in $F$ holds in $T$.

REMARK. Since a functional dependency $X \rightarrow \{A_1, \cdots, A_m\}$ derives functional dependencies $X \rightarrow A_i$, $i=1, \cdots, m$ and vice versa, we restrict our attention to functional dependencies of the form $X \rightarrow A$, where $A$ is a single attribute.

DEFINITION 1.7. An incomplete table $T'$ is said to be an *extension* of an incomplete table $T$ if $T$ and $T'$ have the same attribute set and the same number of rows and each table entry $E$ of $T'$ is included in the corresponding entry of $T$.

DEFINITION 1.8. An incomplete table $T$ is said to be *recoverable* under a family $F$ of funcitonal dependencies if there is an extension of $T$ to a complete table which is consistent with $F$.

EXAMPLE 1.9. Consider the following incomplete table $T$:

| S♯ | PART♯ | QTY | PLACE |
|---|---|---|---|
| S1 | {P1, P2} | 200 | FUKUOKA |
| S1 | P2 | {200, 100} | TOKYO |
| S2 | {P2, P3} | 100 | KYOTO |
| {S1, S2} | P2 | 100 | KOBE |

Then $T$ is recoverable under a functional dependency S♯·PART♯→QTY to the following complete table:

| S♯ | PART♯ | QTR | PLACE |
|---|---|---|---|
| S1 | P1 | 200 | FUKUOKA |
| S1 | P2 | 100 | TOKYO |
| S2 | P2 | 100 | KYOTO |
| S1 | P2 | 100 | KOBE |

However, the following incomplete table is not recoverable under S♯·PART♯→QTY.

| S♯ | PART♯ | QTY | PLACE |
|---|---|---|---|
| S1 | P1 | 100 | FUKUOKA |
| {S1, S2} | P1 | 200 | OSAKA |
| S2 | {P1, P2} | 100 | NAGOYA |
| S2 | P2 | 300 | IWATE |

We are now facing with the problem to decide whether there is at least one recovery of a given incomplete table under a finite collection of functional dependencies. We consider the computational complexity of this problem. For the materials concerning the computational complexity theory, see Aho et al. (1974) and Garey and Johnson (1979). Formally, the problem we consider is

**RECOVERABILITY FOR INCOMPLETE TABLES OF FINITE TYPE**

**INSTANCE:** An incomplete table $T$ of finite type and a finite set $F$ of functional dependencies.

**QUESTION:** Is the incomplete table $T$ recoverable under $F$?

It should be noted that we can check in deterministic polynomial time whether a complete table satisfies a given set of functional dependencies. Therefore we can see that RECOVERABILITY FOR INCOMPLETE TABLES OF FINITE TYPE is solvable in nondeterministic polynomial time just by guessing the elements from entries of a given incomplete table and checking consistency with the functional dependencies.

Furthermore, we can show the following theorem:

THEOREM 1. *RECOVERABILITY FOR INCOMLETE TABLES OF FINITE TYPE is* ***NP****-complete.*

PROOF. For the reduction, we use 3SATISFIABILITY.

**3SATISFIABILITY (3SAT)**

**INSTANCE:** A collection $C=\{c_1, \cdots, c_m\}$ of clauses on a finite set $V$ of variables such that $|c_i|=3$ for $i=1, \cdots, m$.

**QUESTION:** Is there a truth assignment for $V$ that satisfies all clauses in $C$?

Since we have already seen that RECOVERABILITY FOR INCOMPLETE TABLES OF FINITE TYPE is in ***NP***, we show a log space reduction from 3SAT to this problem. Let $C$ be an instance of 3SAT and let $V=\{x_1, \cdots, x_n\}$ be its variable set. We construct an incomplete table and functional dependencies in the following way: The set $U_C$ of attributes consists of literals $x_1, \bar{x}_1, \cdots, x_n, \bar{x}_n$ and all clauses $u_i+v_i+w_i$ in $C$. The set $F_C$ of functional dependencies over $U_C$ contains $x_i \rightarrow \bar{x}_i$ for $i=1, \cdots, n$ and $\{u_i, v_i, w_i\} \rightarrow u_i+v_i+w_i$ for all clauses in $C$. The incomplete table $T_C$ is constructed as follows:

For the first row, the entries corresponding to the attributes $x_i$ and $\bar{x}_i$ for $i=1, \cdots, n$ have the set $\{0, 1\}$. The entries of $u_i+v_i+w_i$ for all clauses have the single value 1.

For dependencies $x_i \rightarrow \bar{x}_i$, we use two rows to represent the negation-function as follows:

| $x_1$ | $\bar{x}_1$ | $x_2$ | $\bar{x}_2$ | $\cdots$ | $x_n$ | $\bar{x}_n$ |
|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 1 | $\cdots$ | 0 | 1 |
| 1 | 0 | 1 | 0 | $\cdots$ | 1 | 1 |

The remaining entries of these two rows corresponding to clauses are filled according to the rule of the or-function and the values of the entries corresponding to literals.

For each dependency $\{u_i, v_i, w_i\} \rightarrow u_i+v_i+w_i$, we use eight rows to represent the orfunction as follows:

| $u_i$ | $v_i$ | $w_i$ | $u_i+v_i+w_i$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

Then we determine the values of entries of literals by using dependencies $x_i \rightarrow \bar{x}_i$, for $i=1, \cdots, n$. The remaining entries of literals of the form $x_j$ and $\bar{x}_j$ are set to be 0 and 1, respectively. The entries for the attributes corresponding to clauses are determined by the or-function and the values of literals in these eight rows.

Then it is easy to see that $C$ is satisfiable iff $T_C$ is recoverable under $F_C$. It is also easy to see that this reduction is done in log space □

Exampxe 1.10. We show the construction in Theorem 1 by an example. Consider the formula $C=(x+y+z)(\bar{x}+\bar{y}+\bar{z})$. The functional dependencies are

$$x \rightarrow \bar{x},\ y \rightarrow \bar{y},\ z \rightarrow \bar{z},$$

$$\{x, y, z\} \rightarrow x+y+z,\ \{\bar{x}, \bar{y}, \bar{z}\} \rightarrow \bar{x}+\bar{y}+\bar{z}.$$

The incomplete table $T_C$ is the following:

| $x$ | $\bar{x}$ | $y$ | $\bar{y}$ | $z$ | $\bar{z}$ | $x+y+z$ | $\bar{x}+\bar{y}+\bar{z}$ |
|---|---|---|---|---|---|---|---|
| {0, 1} | {0, 1} | {0, 1} | {0, 1} | {0, 1} | {0, 1} | 1 | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |

## 2. UNIQUE RECOVERABILITY.

In Section 1 we considered the recoverability of incomplete tables of finite type. For infinite type incomplete tables, we will see that the recoverability can be checked in polynomial time and if an incomplete table is recoverable but not uniquely recoverable, then there are infinitely many recoveries for this incomplete table. On the other hand, for finite type incomplete tables, there are at most finitely many recoveries. The purpose of this section is to study the complexity of the unique recoverability for infinite and finite type incomplete tables.

DEFINITION 2.1. An incomplete table $T$ is said to be *uniquely recoverable under a family $F$ of functional dependencies* if there is exactly one extension of $T$ to a complete table that is consistent with $F$.

We now see the UNIQUE RECOVERABILITY FOR INCOMPLETE TABLES OF INFINITE TYPE is solved in polynomial time. Before showing the procedure which checks the unique recoverability, recall that entries of infinite type incomplete tables are singletons or variables $x[i]$.

The procedure UNIQUE_RECOVERY checks the unique recoverability of infinite type incomplete tables.

```
procedure UNIQUE_RECOVERY ($F$, $T$)
//$F$ is a set of functional dependencies//
//$T$ is an infinite type incomplete table//
  $S \leftarrow$CLOSURE ($F$, $T$);
  while $S \neq T$ do
    $T \leftarrow S$;
    $S \leftarrow$CLOSURE ($F$, $T$)
  end while;
  if $T$ is not consistent with $F$ then return (false);
  if $T$ has a variable then return (false) //recoverable but not//
                                            //uniquely recoverable//
  else return (true) //$T$ contains the unique recovery//
end UNIQUE_RECOVERY
procedure CLOSURE ($F$, $T$)
  for each functional dependency $X \rightarrow A$ in $F$ do
    for each pair $(t_1, t_2)$ of distinct rows of $T$ do
      if $t_1(X)=t_2(X)$ then //$t_i(X)$ is the projection of $t_i$ to $X$//
        case
          : $t_i(A)=x[i]$ and $t_2(A)=x[j]$ for some $i$ and $j$:
                if $i<j$ then $t_2(A) \leftarrow t_1(A)$ else $t_1(A) \leftarrow t_2(A)$
          : $t_1(A)=x[i]$ for some $i$ and $t_2(A)$ is a constant: $t_1(A) \leftarrow t_2(A)$
          : $t_1(A)$ is a constant and $t_2(A)=x[i]$ for some $i$: $t_2(A) \leftarrow t_1(A)$
        end case
    end for
  end for;
  return ($T$)
end CLOSURE
```

The applications of the procedure CLOSURE find the least extension of the given incomplete table under the given functional dependencies, if any. Therefore if the resulting table contains a variable $x[i]$, then the given incomplete table is not uniquely recoverable.

In addition to this polynomial time solution, we can prove the following theorem:

THEOREM 2. *UNIQUE RECOVERABILITY FOR INCOMPLETE TABLES OF INFINITE TYPE is complete for* $\boldsymbol{P}$, *the class of problems solvable in polynomial time, under log space reductions* □

For the proof of Theorem 2, we use the *pebble game on and/or graphs*. An *and/or graph* is a triple $G=(V, E, f)$, where $G'=(V, E)$ is a finite directed graph and $f$ is a function $f: V \rightarrow \{and, or\}$. The rules of the pebble game are the following:

(1) One can always place a pebble on a node which has no incoming edges.

(2) One can place a pebble on node $v$, when $f(v)=and$ and all nodes from which an edge is directed to node $v$ contain pebbles.

(3) One can place a pebble on node $v$, when $f(v)=or$ and at least one node from which an edge is directed to node $v$ contains a pebble.

We use the following problem:

**AND/OR GRAPH ACCESSIBILITY PROBLEM (AGAP)**

**INSTANCE:** An and/or graph $G=(V, E, f)$.

**QUESTION:** Can a pebble be placed on one of the nodes which have no outward edges?

The following theorem is known (Cook (1974), Lingas (1978)).

Theorem 3. *AND/OR GRAPH ACCESSIBILITY PROBLEM is complete for* $\boldsymbol{P}$ *under log space reductions* □

Proof of Theorem 2: Without loss of generality, we may consider and/or graphs having exactly one node which has no outward edges since we can transform in log space the instances of AGAP to and/or graphs of this kind without violating accessibility. Let $G=(V, E, f)$ be an and/or graph which has exactly one outward edge free node. We construct an incomplete table $T_G$ of infinite type and a family $F_G$ of functional dependencies in the following way: Let $V=\{v_1 \cdots, v_n\}$ and assume that $v_n$ is the unique node without any outward edges. The attributes of $T_G$ are $A_1, A_2, \cdots, A_n, B$, where $A_i$ corresponds to the node $v_i$ for each $i=1, \cdots, n$. The attribute $B$ is an auxiliary attribute. The set $F_G$ of functional dependencies consists of the following:

(1) For each node $v_i$ with $f(v_i)=and$, $\{A_j : v_j$ is a predecessor of $v_i\} \rightarrow A_i$ is a functional dependency in $F_G$.

(2) For each node $v_i$ with $f(v_i)=or$, $A_j \rightarrow A_i$ is a functional dependency when $v_j$ is a predecessor of $v_i$.

(3) $A_n B \rightarrow A_i$ is a functional dependency for each $i=1, \cdots, n$.

Let $v_{i_1}, \cdots, v_{i_k}$ be the nodes without any incoming edges and let $v_{j_1}, \cdots, v_{j_{n-k}}$ be the remaining nodes. The incomplete table $T_G$ is the following:

| $A_{i_1}$ | $A_{i_2}$ | $\cdots$ | $A_{i_k}$ | $A_{j_1}$ | $A_{j_2}$ | $\cdots$ | $A_{j_{n-k}}$ | $B$ |
|---|---|---|---|---|---|---|---|---|
| $a$ | $a$ | | $a$ | $x[1]$ | $x[2]$ | $\cdots$ | $x[n-k]$ | $a$ |
| $a$ | $a$ | | $a$ | $a$ | $a$ | | $a$ | $a$ |

where $a$ is a constant value.

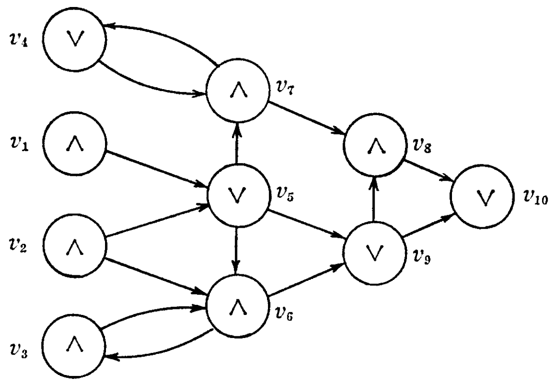
It should be noted that if the node $v_n$ is pebbleable then $T_G$ is uniquely recoverable by using the functional dependencies $A_n B \rightarrow A_i$ for all $i=1, \cdots, n$ and vice versa.

It is easy to see that the set $F_G$ and the incomplete table $T_G$ are computable within log space □

We give an example of construction in Theorem 3.

EXAMPLE 2.2. Consider the following and/or graph $G$. Then the set $F_G$ is

$$\begin{array}{lll}
A_6 \to A_3 & A_2A_3A_5 \to A_6 & A_6 \to A_9 \\
A_7 \to A_4 & A_4A_5 \quad \to A_7 & A_8 \to A_{10} \\
A_1 \to A_5 & A_7A_9 \quad \to A_8 & A_9 \to A_{10} \\
A_2 \to A_5 & A_5 \qquad \to A_9 & A_{10}B \to A_i,\ i=1, \cdots, 10.
\end{array}$$



The table $T_G$ is

| $A_1$ | $A_2$ | $A_3$ | $A_4$ | $A_5$ | $A_6$ | $A_7$ | $A_8$ | $A_9$ | $A_{10}$ | $B$ |
|---|---|---|---|---|---|---|---|---|---|---|
| $a$ | $a$ | $x[1]$ | $x[2]$ | $x[3]$ | $x[4]$ | $x[5]$ | $x[6]$ | $x[7]$ | $x[8]$ | $a$ |
| $a$ | $a$ | $a$ | $a$ | $a$ | $a$ | $a$ | $a$ | $a$ | $a$ | $a$ |

If an incomplete table is shown to be uniquely recoverable, then we may consider the information which is lost is redundant. Recall that the variables appearing in an infinite type incomplete table $T$ are distinct. As long as an incomplete table is uniquely recoverable under a given family of functional dependencies, the larger number of variables means the less redundancy of representation. Thus the maximum number of variables means the minimum redundancy. Then the following problem will receive attention:

Given a complete table $T$ and a family $F$ of functional dependencies, where $T$ is consistent with $F$, then find an incomplete table $T'$ with the maximum number of variables such that $T'$ is uniquely recoverable to the complete table $T$ under $F$.

Formally, we consider the following problem:

**REDUNDANCY MINIMIZATION PROBLEM FOR INFINITE TYPE INCOMPLETE TABLES**

**INSTANCE:** A complete table $T$ with a finite set $F$ of functional dependencies and a nonnegative integer $K$.

**QUESTION:** Is there an incomplete table $T'$ of infinite type such that $T'$ is uniquely recoverable to $T$ under $F$ and the number of variables $x[i]$ is not less than $K$?

Our result asserts that redundancy minimization is computationally hard.

THEOREM 4. *REDUNDANCY MINIMIZATION PROBLEM FOR INFINITE TYPE INCOMPLETE TABLES is* **NP**-*complete.*

PROOF. We reduce the VERTEX COVER PROBLEM (see Garey and Johnson (1979)) to this problem. The VERTEX COVER PROBLEM is stated as follows:

**VERTEX COVER PROBLEM**

**INSTANCE:** An undirected graph $G=(V, E)$ and a positive integer $K \leqq |V|$.

**QUESTION:** Is there a vertex cover of size $K$ or less for $G$, i.e., a subset $V'$ of $V$ with $|V'| \leqq K$ such that for each $\{u, v\}$ in $E$ at least one of $u$ and $v$ belongs to $V'$?

Given an undirected graph $G=(V, E)$ and a positive integer $K$, we construct a complete table $T_G$ over an attribute set $U_G = V \cup E$ and a finite set $F_G$ of functional dependencies in the following way: The complete table over $U_G$ is defined to be a complete table with two rows whose entries are the constant value $a$. The functional dependencies in $F_G$ are as follows:

(1) For each edge $e=\{u, v\}$, the functional dependencies $u \to e$ and $v \to e$ are in $F_G$.

(2) For each node $v$ in $V$, the functional dependency $E \to v$ is in $F_G$.

The integer $K'$ is defined to be $|E|+|V|-K$.

Assume that $G$ has a vertex cover $V'$ of size at most $K$. Let $V'=\{v'_1, \cdots, v'_m\}$, $V-V'=\{v_1, \cdots, v_n\}$ and $E=\{e_1, \cdots, e_p\}$. Then consider the following incomplete table $T'$:

| $v'_1$ | $v'_2 \cdots v'_m$ | | $v_1$ | $v_2 \cdots v_n$ | | $e_1$ | $\cdots$ | $e_p$ |
|---|---|---|---|---|---|---|---|---|
| $a$ | $a$ | $a$ | $x[1]$ | $x[2] \cdots$ | $x[n]$ | $x[n+1]$ | $\cdots$ | $x[n+p]$ |
| $a$ | $a \cdots$ | $a$ | $a$ | $a \cdots$ | $a$ | $a$ | $\cdots$ | $a$ |

By using the functional dependencies of (1) and by the assumption that $V'$ is a vertex cover for $G$, the variables $x[n+1], \cdots, x[n+p]$ are uniquely recoverable. Then by using the functional dependencies of (2), the variables $x[1], \cdots, x[n]$ will be recovered. Observe that the number of variables in $T'$ is $|V|-|V'|+|E| \geqq |V|+|E|-K$.

Conversely assume that there is an incomplete table $T'$ such that $T'$ is uniquely recoverable to $T_G$ under $F_G$ and the number of variables in $T'$ is not less than $|V|+|E|-K$. Note that if there is an attribute such that two entries of this attribute are variables, then the uniqueness of recoverability will be violated. Thus for each attribute, at least one of the entries for the attribute must have the value $a$. We convert $T'$ in the following way: If both entries corresponding to an edge $e=\{u, v\}$ have the value $a$, then we replace the entry of the first row of the attribute $e$ by a new variable and we also replace one of the variables, if any, appearing in columns of the attributes $u$ and $v$ by the value $a$. Let $T''$ be the resulting table. Note that the above conversion does not decrease the number of variables and $T''$ is still uniquely recoverable if $T'$ is. Let $V'$ be the set all attributes whose both of the first and second entries have the value $a$. Then $V' \subseteqq V$ and by the assumption on the number of variables in $T'$, we see that $|V'| \leqq K$. Since all variables in columns for edge attributes are recoverable, $V'$ is a vertex cover for $G$. It is easy to see that $T_G$ and $K'$ are computable in log space. Since UNIQUE RECOVERABILITY FOR INCOM-

PLETE TABLES OF INFINITE TYPE is solvable in deterministic polynomial time, the problem is, in $\mathbf{NP}$. This completes the proof $\square$

By the above theorem, actually to minimize the redundancy is an $\mathbf{NP}$-hard problem.

For the finite type incomplete tables, the situation of unique recoverability is different. We consider the following problem:

**UNIQUE RECOVERABILITY FOR INCOMPLETE TABLES OF FINITE TYPE**

**INSTANCE:** An incomplete table $T$ of finite type and a finite set $F$ of functional dependencies.

**QUESTION:** Is $T$ uniquely recoverable under $F$?

In contrast with the case of infinite type incomplete tables, we have the following theorem:

THEOREM 5. *UNIQUE RECOVERABILITY FOR INCOMPLETE TABLES OF FINITE TYPE is co-$\mathbf{NP}$-hard* $\square$

For the proof of Theorem 5, we use the following problem, which will be shown to be co-$\mathbf{NP}$-hard.

**UNIQUE 4SATISFIABILITY (U4SAT)**

**INSTANCE:** A finite collection of clauses $C$ on some variable set $V$, where for each clause $c$ in $C$, $|c|=4$.

**QUESTION:** Is $C$ uniquely satisfiable, i.e., is there only one truth assignment which satisfies all clauses in $C$?

LEMMA 6. *UNIQUE 4SATISFIABILITY is co-$\mathbf{NP}$-hard.*

PROOF. We show a reduction from 3SAT to co-U4SAT, the complement of U4SAT. Let $C$ be an instance of 3SAT and let $V=\{x_1, \cdots, x_n\}$ be its variable set. We construct a new set of clauses in the following way: Let $x_{n+1}$, $x_{n+2}$, $x_{n+3}$, $x_{n+4}$, $x_{n+5}$, and $x_{n+6}$ be new variables not in $V$. Let $C=\{c_1, \cdots, c_m\}$ and let $c_i=u_i+v_i+w_i$ for $i=1, \cdots, m$. For each $i=1, \cdots, m$, we use the clause $c'_i=u_i+v_i+w_i+x_{n+1}$. In addition to these clauses, we use the clauses $c'_{n+j}=\bar{x}_j+\bar{x}_{n+1}$ for $j=1, \cdots, n$. Then $C'$ consists of all such $c'_j$.

Note that the assignment $(x_1, \cdots, x_{n+1})\leftarrow(0, \cdots, 0, 1)$ satisfies all clauses in $C'$. Thus $C'$ has at least one satisfiable assignment. If $C$ is satisfiable, then let $(x_1, \cdots, x_n)\leftarrow(a_1, \cdots, a_n)$ be one of the truth assignments that satisfy $C$. Then we can see that the truth assignment $(x_1, \cdots, x_{n+1})\leftarrow(a_1, \cdots, a_n, 0)$ also satisfies all clauses in $C'$. Therefore, there are more than one satisfiable assignments for $C'$. Conversely, assume that $C'$ has at least two truth assignments that satisfy all clauses in $C'$. Let $(x_1, \cdots, x_{n+1})\leftarrow(a_1, \cdots, a_{n+1})$ be one of such truth assignments. If $a_{n+1}=1$, then $a_1=\cdots=a_n=0$. If $a_{n+1}=0$, then $(x_1, \cdots, x_n)\leftarrow(a_1, \cdots, a_n)$ must be a truth assignment that satisfies $C$. Therefore, $C$ is satisfiable iff $C'$ is not uniquely satisfiable. Finally, we replace the clauses $\bar{x}_i+\bar{x}_{n+1}$ by $\bar{x}_i+\bar{x}_{n+1}+\bar{x}_{n+2}+\bar{x}_{n+3}$ for all $i=1, \cdots, n$. In order to force $\bar{x}_{n+2}=\bar{x}_{n+3}=0$, we add to $C'$ the clauses $x_k+u+v+w$, where $k=n+2$, $n+3$, and $u=x_{n+4}$, $\bar{x}_{n+4}$, $u=x_{n+5}$, $\bar{x}_{n+5}$, $w=x_{n+6}$, $\bar{x}_{n+6}$ $\square$

Proof of Theorem 5 can be done by reducing U4SAT to UNIQUE RECOVERABILITY FOR INCOMPLETE TABLES OF FINITE TYPE in a similar manner to Theorem 1 except that in this case four argument or-function is used $\square$

As an upper bound, it is not hard to see that this problem can be solved by a deterministic polynomial time bounded oracle Turing machine that has RECOVERABILITY FOR INCOMPLETE TABLES OF FINITE TYPE as its oracle. Thus this problem is in $\Delta_2^p$, the class of problems solvable in deterministic polynomial time oracle Turing machines with oracles in $\boldsymbol{NP}$ (Stockmeyer (1977)).

## 3. RECOVERY OF TIME-VARIANT TABLES

In this section we will deal with time-variant tables. We use "day" for time unit. Informally, a time-variant table is a two-way infinite, past and future, sequence of tables with the same attribute set and the same number of rows. For this time-variant table, we will pay attention to the time-invariant properties of "everyday" and the time-invariant relation between "yesterday" and "today". This time-invariant relation is expressed by functional dependencies over the attribute sets of "yesterday" and "today".

NOTATION. Let $U=\{A_1, \cdots, A_n\}$ be a finite set of attributes. We mean by $U^\infty$ the set $\{A_j(t): 1 \leqq j \leqq n$, and $t$ is an integer$\}$. The attribute $A_j(t)$ will denote the occurrence of the attribute $A_j$ at time $t$. The set $U^{(t)}$ is defined to be $\{A_j(t): 1 \leqq j \leqq n\}$ for each $t$. For each attribute $A_j$, we assume that $D(A_j)=D(A_j(t))$ for all $t$.

DEFINITION 3.1. Let $U$ and $U^\infty$ be as above. *A time-variant incomplete table over an attribute set U* is an incomplete table over $U^\infty$. We define in the same way time-variant complete tables over $U$.

DEFINITION 3.2. Let $X^{(0)}=\{A_{i_1}(0), \cdots, A_{i_m}(0), A_{j_1}(1), \cdots, A_{j_k}(1)\}$ and let $A^{(0)}$ be $A_j(i)$, where $i=0$ or 1. Given a functional dependency $X^{(0)} \rightarrow A^{(0)}$, we define the *dynamic expansion of* $X^{(0)} \rightarrow A^{(0)}$ as a family of functional dependencies $\{X^{(t)} \rightarrow A^{(t)}: t$ is an integer$\}$, where for each integer $t$ we define $X^{(t)}=\{A_{i_1}(t), \cdots, A_{i_m}(t), A_{j_1}(t+1), \cdots, A_{j_k}(t+1)\}$ and $A^{(t)}=A_j(i+t)$. In the same way we define the *dynamic expansion* $F^\infty$ when we are given a family $F$ of functional dependencies over $U^{(0)} \cup U^{(1)}$.

DEFINITION 3.3. Given an incomplete table $T$ over $U$, we define an incomplete time-variant table $T^\infty$ over $U$ called the *dynamic expansion of* $T$ by repeating $T$ at each time. More formally, $T^\infty$ has the same number of rows and for each attribute $A_j$ in $U$, $(r, A_j)$-entry of $T$ is equal to $(r, A_j(t))$-entry of $T^\infty$ for each $t$ and each row $r$.

DEFINITION 3.4. Let $F$ be a family of functional dependencies over $U^{(0)} \cup U^{(1)}$ and let $T$ be a time-variant complete table over $U$. Then we say that *the family F holds for T dynamically* if $T$ is consistent with $F^\infty$.

DEFINITION 3.5. Given an incomplete table $T$ over $U$ and a family $F$ of functional dependencies over $U^{(0)} \cup U^{(1)}$, $T$ is said to be *dynamically recoverable under F* if the dynamic expansion $T^\infty$ of $T$ is recoverable under the dynamic expansion $F^\infty$ of $F$.

We are interested in the following recovery problem:

**DYNAMIC RECOVERABILITY FOR INCOMPLETE TABLES OF FINITE TYPE**

**INSTANCE:** An incomplete table $T$ over an attribute set $U$ and a finite collection $F$ of functional dependencies over $U^{(0)} \cup U^{(1)}$.

**QUESTION:** Is $T$ dynamically recoverable under $F$?

We can regard the values of entries having a single element as the values which never change and we can also deal with the entries with multiple elements as time-variant entries.

We will show the following theorem:

THEOREM 7. *DYNAMIC RECOVERABILITY FOR INCOMPLETE TABLES OF FINITE TYPE is complete for* ***PSPACE****, the class of problems solvable in polynomial space, under log space reductions* □

We first see that this dynamic recovery problem is in ***PSPACE.***

```
prcedure DYNAMIC_RECOVERY ($F$, $T$)
//$T$ is an incomplete table over an attribute set $U$//
//$F$ is a family of functional dependencies over $U^{(0)} \cup U^{(1)}$//
  guess a complete table $T_0$ from $T$; $T_1 \leftarrow T_0$;
  repeat
    guess a complete table $T_2$ such that $T_1T_2$ is consistent with $F$;
    $T_1 \leftarrow T_2$
  until $T_2T_0$ is consistent with $F$;
  return (true)
end DYNAMIC_RECOVERY
```

Note that if the given incomplete table is dynamically recoverable under $F$, then there are two points of time $i<j$ such that the complete table $T^{(i)}$ of the restriction to the time $i$ of the dynamically recovered complete table is the same as that of the time $j$ since there are only finitely many extensions of $T$. Conversely, if there exists a sequence of complete tables $T_0, T_1, \cdots, T_m$ that are extensions of $T$ such that $T_0=T_m$ and each $T_iT_{i+1}$ is consistent with $F$, then $T$ is easily seen to be dynamically recoverable under $F$.

By the above observation, we can see that the procedure DYNAMIC_RECOVERY runs correctly. Obviously, the space amount used in this procedure is polynomial. By Savitch's theorem (see Aho et al. (1974)), the class of problems solvable using nondeterministic polynomial space is the same as the deterministic class. Thus DYNAMIC RECOVERABILITY FOR INCOMPLETE TABLES OF FINITE TYPE is in ***PSPACE.***

For the proof of Theorem 7, We use DYNAMIC 3SATISFIABILITY, which was shown to be complete for ***PSPACE*** under log space reductions (Orlin (1981)).

**DYNAMIC 3SATISFIABILITY (DYNAMIC 3SAT)**

**INSTANCE:** A collection $C=\{c_1, \cdots, c_m\}$ of clauses over $\{x_1(0), \cdots, x_n(0), x_1(1), \cdots, x_n(1)\}$ such that each clause $c_i$ in $C$ has $|c_i|=3$.

**QUESTION:** Is $C$ dynamically satisfiable? That is, we define the set $C^{(i)}$ as a collection of clauses obtained from the clauses in $C$ by replacing literals $v_j(\sigma)$ by $v_j(\sigma+i)$ and set $C^{\infty}$ to be the union of $C^{(i)}$, where $i$ ranges over all integers. Then is there truth assignment for variables in $\{x_j(i): 1 \leqq j \leqq n,$ and $i$ is an integer$\}$ which satisfies all clauses in $C^{\infty}$?

We now show a reduction from DYNAMIC 3SAT to DYNAMIC RECOVERABILITY FOR INCOMPLETE TABLES OF FINITE TYPE.

Let $C$ be an instance of DYNAMIC 3SAT. We assume that the underlying variable set is $\{x_1, \cdots, x_n\}$. We construct the family $F_C$ of functional dependencies and the incomplete table $T_C$ in the following way: The set $U_C$ of attributes for $T_C$ consists of $x_1(0), \cdots, x_n(0), \bar{x}_1(0), \cdots, \bar{x}_n(0), x_1(1), \cdots, x_n(1), \bar{x}_1(1), \cdots, \bar{x}_n(1),$ and clauses $v_i(\sigma)$

$+v_j(\tau)+v_k(\rho)$ in $C$.

The functional dependencies in $F_C$ over $U^{(0)} \cup U^{(1)}$ are the following:

(1) $x_i(\sigma)[0] \rightarrow \bar{x}_i(\sigma)[0]$ for $i=1, \cdots, n$ and $\sigma=0, 1$.

(2) $\{v_i(\sigma)[0], v_j(\tau)[0], v_k(\rho)[0]\} \rightarrow (v_i(\sigma)+v_j(\tau)+v_k(\rho))[0]$ for all clauses $v_i(\sigma)+v_j(\tau)+v_k(\rho)$ in $C$.

(3) $x_i(0)[1] \rightarrow x_i(1)[0]$ for $i=1, \cdots, n$.

The functional dependencies of (1) and (2) are meant to represent the negation-function and the or-function, respectively. The functional dependencies of (3) are used to connect "yesterday" and "today" by the same values. The rows of the incomplete table $T_C$ over $U_C$ are constructed as follows: For the first row, the entries of $x_j(\sigma)$ and $\bar{x}_j(\sigma)$ have the set $\{0, 1\}$. The entries of the clauses $v_i(\sigma)+v_j(\tau)+v_k(\rho)$ in $U_C$ have 1. The next two rows are used to guarantee that the entries corresponding to $x_j(\sigma)$ and $\bar{x}_j(\sigma)$ of the same row do not have the same value as we did in the proof of Theorem 1. For each functional dependency of (2), we use eight rows to define or-function and we fill these entries in the same way as in Theorem 1. Finally, last two rows are used to represent the identity function to which the functional dependencies of (3) correspond. That is, for each pair of $x_j(0)$ and $x_j(1)$, we set

| $x_j(0)$ | $x_j(1)$ |
|---|---|
| 0 | 0 |
| 1 | 1 |

The remaining entries are determined in the same way as in Theorem 1 so that it will be consistent with functional dependencies of (1) and (2).

Then we can see that $C$ is dynamically satisfiable if $T_C$ is dynamically recoverable under $F_C$. It is not hard to see that this transformation can be done using log space □

EXAMPLE 3.6. Consider the formula $C=(x(0)+\bar{x}(1)+y(0))$. Then $F_C$ consists of $x(0)[0] \rightarrow \bar{x}(0)[0]$, $y(0)[0] \rightarrow \bar{y}(0)[0]$, $x(1)[0] \rightarrow \bar{x}(1)[0]$, $y(1)[0] \rightarrow \bar{y}(1)[0]$, $x(0)[1] \rightarrow x(1)[0]$, $y(0)[1] \rightarrow y(1)[0]$, and $\{x(0)[0], \bar{x}(1)[0], y(0)[0]\} \rightarrow (x(0)+\bar{x}(1)+y(0))[0]$.

The incomplete table $T_C$ is:

| $x(0)$ | $\bar{x}(0)$ | $y(0)$ | $\bar{y}(0)$ | $x(0)+\bar{x}(1)+y(0)$ | $x(1)$ | $\bar{x}(1)$ | $y(1)$ | $\bar{y}(1)$ |
|---|---|---|---|---|---|---|---|---|
| {0, 1} | {0, 1} | {0, 1} | {0, 1} | 1 | {0, 1} | {0, 1} | {0, 1} | {0, 1} |
| 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 |

REMARK. As in Section 2, we may consider the unique dynamic recoverability for infinite type incomplete tables. However, it is not difficult to show that the complexity of the unique dynamic recoverability is the same as that shown in Section 2.

## 5. CONCLUDING REMARKS AND FURTHER RESEARCH

In this paper we showed that in general the recovery problems are computationally hard. However, the problem of recovery refuses to vanish at the sound of this fascinating word. How can we cope with this problem ? One approach to this proplem is to use the idea of approximation algorithm. We believe that we can discuss the recovery problem in the following setting and apply the techniques from approximation algorithms.

Each information has some cost. The problem is: Given an incomplete information $X$, then recover the incomplete information $X$ to a complete information $Y$ so as to minimize or maximize the cost of $Y$.

With respect to the above problem, there should be an assumption that every incomplete information is at least easily recovered to a complete information and it is not a good idea to stick to "recoverability" as is discussed in this paper.

Another approach we would like to propose is the recovery with knowledge bases, that is, to recover the lacking information using some knowlege bases. Through the help of knowlege bases, we should gain the computational easiness of recovery procedures in addition to the advantage for finding lost information. For this idea, we may consider polynomial time bounded oracle Turing machines as a model of recovery machines. Oracles here are considered as knowledge bases and polynomial means computational easiness. In Section 1, the instances of the recovery problem are incomplete tables and functional dependencies. We can modify this problem as follows: The incomplete tables are used in the same way. But each functional dependency is associated with a function specification symbol. As a knowledge base, we have a collection of functions that correspond to the function specification symbols. A recovery procedure makes queries to the knowledge base. We think this approach is natural. However, at present, we do not have any real examples appropriate for this framework and we have not analysed what problems will arise.

In Section 2, we showed that the unique recoverability problem for incomplete tables of infinite type is solvable in polynomial time but the problem for finite type incomplete tables remains hard. We also considered the redundancy minimization problem. We believe that problems arising from incomplete information recovery are deeply related with information redundancy. From this point of view, there can be a research which deals with the connection between the information recovery and the information redundancy.

Throughout this paper, we were concerned with the relational model of data and functional dependency. However, there is no reason why we must remain in this model and in this dependency. We just chose and studied the relational model and functional dependency as a first step toward the theory of incomplete information recovery. In literatures, various kinds of dependencies for relational model are introduced, for instance, multivalued dependency, mutual dependency, and transitive dependency (Paredaen (1980)). We are now in the process of studying these dependencies in connection with incomplete information recovery.

Furthermore, we can regard the spelling correction problem (Peterson (1980)) as one of the incomplete information recovery problems. In this case, information incompleteness means "error". There are several kinds of errors occurring in spelling correction, for example transmission error, author ignorance, typographical error etc. What we should do for this problem is to find a "best possible" text from a given text which possibly contains errors by using the knowledge about error generation processes and the context of the text. Usually edit-distance or something similar to this is used as a measure of "best possibility" (Kashyap and Oommen (1981)). By using measures of this kind, we may consider optimal spelling correctors. As is indicated in this spelling correction problem, we need a powerful theory which can deal with recovery with optimality in nonnumeric data processing. This possibility is suggested in Traub and Wazniakowski (1981). We think there are many problems to be studied in this field.

## ACKNOWLEDGEMENTS

The authors would like to express our gratitude to colleagues at Research Institute of Fundamental Information Science at Kyushu University for helpful discussion. We would also thank Prof. Sueo Iimori for sending us some useful papers.

## References

[1] AHO, A. V., J. E. HOPOROFT, and J.D. Ullman (1974), *The Design and Analysis of Computer Algorithms*, Addison-Wesley.

[2] CODD, E. F. (1970), *A relational model of data for large shared data banks*, Comm. ACM **13**, 377-397.

[3] CODD, E. F. (1971), *Further normalization of the database relational model*, Courant Computer Science Symposia **6**, "Data Base Systems," New York, Prentice-Hall, 33-64.

[4] COOK, S. A. (1974), *An obserbation on time-storage trade-off*, J. Comput. System Sci. **9**, 308-316.

[5] GAREY, M. R. and D. S. JOHNSON (1979), *Computers and Intractability*, W.H. Freeman and Company, San Francisco.

[6] JAEGERMANN, M. (1978, 1979), *Information storage and retrieval systems with incomplete information* I II, Fundamenta Inform. **2**, 17-41, 141-166.

[7] KASHYAP, R. L. and B. J. OOMMEN (1981), *An effective algorithms for string correction using generalized edit distances*, Inform. Sci. **23**, 123-142.

[8] LINGAS, A. (1978), *A PSPACE-complete problem related to a pebble game*, Proc. of the Fifth Colloquium on Automata, Languages and Programming (Lecture Notes in Computer Science **62**) G. Ausiello and C. Böhm, Ed., 300-321.

[9] LIPSKI, W. JR. (1979), *On semantic issues connected with incomplete information databases*, ACM Trans. on Database Systems **4**, 262-296.

[10] LIPSKI, W. JR. (1981), *On data bases with incomplete information*, J. Assoc. Comput. Mach. **28**, 41-70.

[11] NAKAMURA, A. (1981), *On the monadic predicate logic of incomplete information*, Project Reprot "Knowledge Representations and Their Application to Information Retrieval," S. Arikawa, Ed., 291-300.

[12] ONO, H, and A. NAKAMURA (1980), *Decidability results on a query language for data bases with incomplete information*, Proc. of the Ninth Int. Symp. on Mathematical Foundations of Computer Science, P. Dembinski, Ed., Springer-Verlag, 452-459.

[13] ORLIN, J. B. (1981), *The complexity of dynamic languages and dynamic optimization problems*, Proc. of the 13th ACM Symp. on Theory of Computing, 218-227.

[14] PAREDAEN, J. (1980), *Transitive dependecies in a database scheme*, RAIRO Informatique/Computer Science **14** n° 2, 149-163.

[15] PETERSON, J. L. (1980), *Computer Programs for Spelling Correction*, Lecture Notes in Computer Science **96**, Springer-Verlag.

[16] STOCKMEYER, L. (1977), *The polynomial-time hierarchy*, Theoret. Comput. Sci. **3**, 1-22.

[17] TRAUB, J. F. and H. WAZNIAKOWSKI (1980), *A General Theory of Optimal Algorithms*, Academic Press.

[18] VASSILIOU, Y. (1980), *Functional dependencies and incomplete information*, Proc. of Int. Conf. on Very Large Data Bases, 6th, 260-269.

*Communicated by S. Arikawa*