JOURNAL OF APPLIED MATHEMATICS AND DECISION SCIENCES, 7(1), 49–59

Discovery of Functional and Approximate Functional Dependencies in Relational Databases

RONALD S. KING† rking@mail.uttyl.edu
Computer Science Department, The University of Texas at Tyler, Tyler, Texas 75799

JAMES J. LEGENDRE
Marathon Oil, Houston, Texas

Abstract. This study develops the foundation for a simple, yet efficient method for uncovering functional and approximate functional dependencies in relational databases. The technique is based upon the mathematical theory of partitions defined over a relation's row identifiers. Using a levelwise algorithm the minimal non-trivial functional dependencies can be found using computations conducted on integers. Therefore, the required operations on partitions are both simple and fast. Additionally, the row identifiers provide the added advantage of nominally identifying the exceptions to approximate functional dependencies, which can be used effectively in practical data mining applications.

Keywords: Functional Dependencies, Data Mining, Approximate Functional Dependencies

1. Introduction

The complexity of discovering functional dependencies has been studied in [5], [6], [7]. Functional dependencies are relationships between attributes of a database relation. A functional dependency states that the value of an attribute is uniquely determined by the value of some other attributes. Algorithmic approaches to the discovery of functional dependencies has been studied in [2], [6], [10], [11], [3].

Suppose that a company sets up a database to keep track of its employees and the various departments to which they are assigned from time to time. This would require three relations: one for employees, one for departments and one for assignments of employees to departments. An instance of this database might include the following relations:

† Requests for reprints should be sent to Ronald King,Computer Science Department, The University of Texas at Tyler, Tyler, Texas 75799.

Table 1. Company Database

Employee number	First name	Family name	Date joined	Salary
123	David	Jones	15-Feb-1995	$32,000
234	Alice	Johnson	4-Mar-1995	$36,000
345	Mary	Trent	31-Aug-1990	$44,000

Department code	Department name
DEV	Development
SLS	Sales
SPT	Support
MKT	Marketing

Employee number	Department code	Date assigned
123	DEV	15-Feb-1995
123	MKT	1-Dec-1998
234	SLS	4-Mar-1998
345	MKT	31-Aug-1990
345	SLS	5-Jun-1994
345	SPT	18-Oct-1998

In the ASSIGNMENTS relation Employee number does not functionally determine Date Assigned, but Employee number plus Department code does functionally determine Date assigned. Employee number is a primary key for the EMPLOYEES relation since the Employee number uniquely functionally determines all of the remaining attributes within the relation. Department code is the primary key for the DEPARTMENTS relation.

An approximate functional dependency is a functional dependency that almost holds. In a relation a few rows can contain errors, due to various noise factors, or simply be a row that is an exception to the rule. Many operational definitions for approximate functional definitions have been studied [4]. The definition utilized in this paper is based upon the minimum number of rows that need to be removed from the relation r for $X \rightarrow A$ to hold in r: the error, $g_3(X \rightarrow A) = 1 - (max\{|s| : s \subseteq r$ and $X \rightarrow A$ holds in s$\}/|r|$. $X \rightarrow A$ is an approximate dependency if and only if $g_3(X \rightarrow A) \leq \epsilon$ for $0 \leq \epsilon \leq 1$. If the following modification for the EMPLOYEES relation is made: Then First name plus Family name does not functionally determine Salary since the first entry has a Salary

Table 2. Modified EMPLOYEES Relation: Approximate Functional Dependency $Firstname \| Familyname \rightarrow Salary$

Employee number	First name	Family name	Date joined	Salary
123	David	Jones	15-Feb-1995	$32,000
234	Alice	Johnson	4-Mar-1995	$36,000
345	Mary	Trent	31-Aug-1990	$44,000
127	David	Jones	5-Jan-2001	$33,000
128	David	Jones	4-Mar-2000	$33,000
129	David	Jones	23-Jan-2000	$33,000

value that is not equal to the Salary value for the last three entries in the relation. Note that First name plus Family name determining Salary is an approximate functional dependency with a 25 percent error rate.

The algorithm for discovering functional and approximate functional dependencies employed in this paper is similar to the levelwise approach for the discovery of association rules [1]. This search strategy first computes some non–trivial information about attribute sets, frequent item sets, and then which association rules can be computed easily. In the present study, first computations for the non–trivial information about attribute sets takes the form of partitions of row identification numbers, from which the dependencies can be computed. [8] employed the levelwise method for the computations of dependencies as an instance of the generic data mining algorithm. [9] introduced the concept of rough sets which is based upon partitions. Using rough sets [12] utilized rough sets for identifying the most critical factors for allowing for the elimination of irrelevant attributes in a relation prior to the generation of rules describing data dependencies in databases.

2. Functional Dependencies and Partitions

Definition. Rows s and t are equivalent with respect to a set of attributes X if and only if $s[A] = t[A]$ for all $A \varepsilon X$.

Note that the definition for equivalent rows on a set of attributes X partitions the rows of the relation into equivalence classes. The equivalence class of a row $t \varepsilon r$ with respect to a given set $X \subseteq R$ by $[t]_X$. The set $\Pi_X = \{[t]_X | t \varepsilon r\}$ of equivalence classes is a partition of r under X.

THEOREM 1 *A functional dependency $X \rightarrow A$, where A is a single attribute and X is a set of attributes, holds if and only if Π_X refines Π_A.*

Proof: Case I: Assume that Π_X refines Π_A.
Let $t, u \; \varepsilon \; R$ where $t[X] = u[X]$. Then $t, u \; \varepsilon \; \Pi_X$

$\Rightarrow u, t \; \varepsilon \; C$ where $C \; \varepsilon \; \Pi_X$

$\Rightarrow u, t \; \varepsilon \; D$ where $D \; \varepsilon \; \Pi_A$

$\Rightarrow u[A] = t[A]$

Thus $u[X] = t[X] \Rightarrow u[A] = t[A] \Rightarrow (X \rightarrow A)$.
Case II: Suppose $X \rightarrow A \; t[X] = u[X] \Rightarrow t[A] = u[A]$.
Let $C \varepsilon \Pi_X$. Then C is a set of t"s and u"s such that $t[X] = u[X]$. But this implies that $t[A] = u[A]$. Therefore, since $X \rightarrow A$, $t[A] = u[A]$ which implies that $C \subseteq D$ where D is a set of t"s and u"s such that $t[A] = u[A]$. Thus there exists $C \subseteq D$ such that $D \varepsilon \Pi_A$. We can therefore conclude that Π_X refines Π_A. □

An extremely interesting simplification for the latter theorem exists which states that adding the attribute A to the set of attributes X does not break any equivalence classes of Π_X whenever Π_X refines Π_A.

THEOREM 2 *$X \Rightarrow A$ if and only if $|\Pi_X| = |\Pi_{X \bigcup \{A\}}|$, where $|\Pi_X|$ denotes the rank of the partition Π_X (or the number of equivalence classes belonging to the partition).*

Proof: Case I: Assume that $X \rightarrow A$. Then adding A to X does not break any equivalence classes in X, since $t[A] = u[X] \Rightarrow t[A] = u[A]$. Thus $\Pi_{X \bigcup \{A\}} = \Pi_X$. But this requires that $|\Pi_X| = |\Pi_{X \bigcup \{A\}}|$.
Case II: Suppose that $|\Pi_X| = |\Pi_{X \bigcup \{A\}}|$. Note that $\Pi_{X \bigcup \{A\}}$ always refines Π_X. Let $u, t \varepsilon C$ where $C \varepsilon \Pi_X$, since $u[X] = t[X]$. But $C = D$ where $D \varepsilon \Pi_{X \bigcup \{A\}}$. Therefore $t[XA] = u[XA]$. But $t[XA] = u[XA] \Rightarrow t[A] = u[A]$. Thus we have demonstrated that $t[X] = u[X] \Rightarrow t[A] = u[A]$ or $X \rightarrow A$. □

Using Theorem 2 one can determine the approximate functional dependencies for a relation r. The error $g_3(X \rightarrow A)$ for a functional dependency, $X \rightarrow A$, is the minimum fraction of rows that must be removed from the relation for the dependency to hold. Note that any equivalence class C of Π_X is the union of one or more equivalence classes C'_1, C'_2, ... of $\Pi_{X \bigcup \{A\}}$, and the rows in all but one of the C'_i s must be removed for $X \rightarrow A$ to be valid. The minimum number of rows to remove is the size of C minus the size of the largest of the C'_i s. Therefore,
$g_3(X \rightarrow A) = 1 - \sum_{C \varepsilon \Pi_X} max\{|C'| : C' \varepsilon \Pi_{X \bigcup \{A\}} \text{ and } C' \subseteq C\}/|r|$.
Clearly any superkey has the property that its' partition consists of singleton equivalence classes only. Additionally, a set X is a key it is a superkey

and no proper subset of it is a superkey. These observations lead to the following definition:

Definition. The error of a superkey, $g_3(X)$, is the minimum fraction of rows that need to be removed from the relation r for X to be a superkey. Given an error threshold ϵ, where $0 \leq \epsilon \leq 1$, then X is an approximate superkey if and only if $g_3(X)$ is at most ϵ.

The partition Π_X can be utilized for computing $g_3(X)$: $g_3(X) = 1 - |\Pi_X|/|r|$. Also, due to the latter computation, we have:

LEMMA 1 *$g_3(X) = g_3(Y)$ if and only if $|\Pi_X| = |\Pi_Y|$.*

Then from Theorem 2 we have:

THEOREM 3 *$X \rightarrow A$ if and only if $g_3(X) = g_3(X \bigcup \{A\})$.*

The latter foundation can be employed in data mining to find all minimal non–trivial functional dependencies by searching through the space of non-trivial dependencies and testing the validity and minimality for each dependency. A functional dependency $X \rightarrow A$ for which there does not exist $Y \subset X$ such that $Y \rightarrow A$ is called a minimal functional dependency. But these minimal non-trivial dependencies are test for validity by taking refinements of partitions and superkeys are represented by partitions containing only singleton equivalence classes. This leads to the observation that a singleton equivalence class, of the left-hand side of a functional dependency, cannot break any dependency.

3. Optimizations via Constrained Partitions

Both space conservation and efficiency consideration lead to the concept of constrained partitions.

Definition. A partition with singleton equivalence classes removed is called a constrained partition. $\bar{\Pi}$ denotes the constrained partition for the partition Π.

Theorem 1 still holds for constrained partitions, since the refinement relationships of partitions are not affected by the singleton equivalence classes. But $| \bar{\Pi}_X |$ can be the same as $| \bar{\Pi}_{X \bigcup \{A\}}|$ even if $|\Pi_X| = |\Pi_{X \bigcup \{A\}}|$. However, Theorem 2 can be employed. The value $g_3(X)$ can be found using constrained partitions:

$g_3(X) = (|| \bar{\Pi}_X || - | \bar{\Pi}_X |/|r|)$,

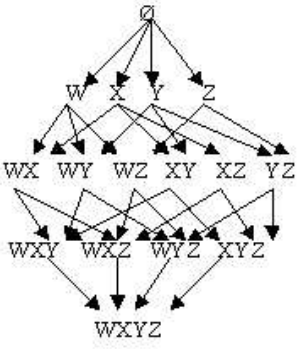

Figure 1. The set containment lattice for W, X, Y, Z:

where $||\ \bar{\Pi}_X\ ||$ is the sum of the cardinalities of the equivalence classes in $\bar{\Pi}_X$. The computing error $g_3(X \rightarrow A)$ for the relation r is $O(|r|)$. But employing the operational definitions for $g_3(X \rightarrow A)$ and $g_3(X)$ leads to:

$$g_3(X) - g_3(X \bigcup \{A\}) \leq g_3(X \rightarrow A) \leq g_3(X)$$

Therefore, if $g_3(X) - g_3(X \bigcup \{A\}) \geq \varepsilon$ or $g_3(X) < \epsilon$, then one does not need to compute $g_3(X \rightarrow A)$ to find whether or not $X \rightarrow A$.

4. Searching for the Non-Trivial Minimal Functional Dependencies

Partitions of row numbers can be utilized to perform necessary validity tests on functional dependencies. Computations of constrained partitions make these validity tests able to be done efficiently.

The search for the functional / approximate functional dependencies consist of the space of all left–hand sides of potential dependencies. For example the set containment lattice, the latter space, for W, X, Y, Z is illustrated in Figure 1.

Using a levelwise algorithm the search starts from the singleton sets, and works its way through the lattice level by level until the minimal functional dependencies are found. For each set X of attributes, the algorithm will test dependencies of the form $X - \{A\} \rightarrow A$, where $A \varepsilon X$. False dependen-

cies are eliminated as early as possible in order to reduce the search space. An edge between sets X and $X \bigcup \{A\}$ in the containment set lattice represents the non-trivial dependency $X \rightarrow A$. Search efficiency is obtained by reducing the computations on each level by using results from previous level computations.

By using the lattice, one can compute a partition as a product of two earlier partitions. The product of two partitions Π and Π', denoted by $\Pi \times \Pi'$, is the least refined partition Π'' that refines both Π and Π'.

THEOREM 4 $\Pi_X \times \Pi_Y = \Pi_{X \bigcup Y}$ *for all* $X, Y \subseteq R$.

Proof: Case I: Assume we are given Π_X and Π_Y. For each $C \varepsilon \Pi_X$ form the set of all nonempty intersections which can be formed from the equivalence classes of Π_Y with C, call this set C_X. Then the union of all C_X's is by construction the least refined partition for both Π_X and Π_Y. Therefore the latter construction yields $\Pi_X \times \Pi_Y$. Let $u, t \varepsilon C$ for some $C \varepsilon \Pi_X \times \Pi_Y$. Then there exist a $D \varepsilon \Pi_X$ and there exist an $E \varepsilon \Pi_Y$ such that $t[X] = u[Y]$ and $t[Y] = u[Y]$ where $C \subseteq D$ and $C \subseteq E$. Thus $C \subseteq D \bigcap E$ where $t[X \bigcup Y] = u[X \bigcup Y]$. Thus $\Pi_X \times \Pi_y$ is a refinement of $\Pi_{X \bigcup Y}$.
Case II: $t, u \varepsilon C$ for some $C \varepsilon \Pi_{X \bigcup Y} \Rightarrow t[X \bigcup Y] = u[X \bigcup Y]$. Then $t, u \varepsilon C$ where $t[X] = u[X]$ and $t[Y] = u[Y]$. Thus by construction there exist a $D \varepsilon \Pi_X \times \Pi_Y$ where $t[X] = u[X]$ and $t[Y] = u[Y]$. Therefore, $\Pi_{X \bigcup Y}$ is a refinement of $\Pi_X \times \Pi_Y$. Since we have set containment in both directions we have $\Pi_{X \bigcup Y} = \Pi_X \times \Pi_Y$. □

A levelwise computational scheme can be employed to compute the partitions. First the computations for $\Pi_{\{A\}}$, for each attribute $A \varepsilon R$, are performed directly on the database. Partitions Π_X, for $|X| \geq 2$, are computed as the product of partitions with respect to two subsets of X. To find a size two partition the direct product of two singleton set partitions will be employed. A size k partition will be found using the direct product of a $(k-1)$ and a singleton partition, thus only the partitions from the previous levels are employed in the computations for finding partitions on the present level.

Once the partition Π_X is found, the error $g_3(X)$ is computed in order to test a functional dependencies validity using Theorem 3. The same value $g_3(X)$ can be utilized for testing the validity of $X \rightarrow A$ or $X - \{A\} \rightarrow A$ for several $A \varepsilon R$.

5. Testing for Minimality of Functional Dependencies

To test for the minimality of $X - \{A\} \rightarrow A$, we need to know whether $Y - \{A\} \rightarrow A$ for some proper subset Y of X.

Definition. Let $C(X-\{A\})$ be the set of right-hand candidates of $X-\{A\}$ for all A.

$A\varepsilon C(X)$ if A has not been found to depend on any proper subset of X. The set of right-hand-candidates (rhs) of a set $X \subseteq R$ is $C(X) = R-C(X)'$, where $C(X)' = \{A\varepsilon X : X - \{A\} \to A\}$.

Assume that the proposed algorithm is considering the set $X = \{A, B, C\}$, and that $\{C\} \to A$. Since $\{C\} \to A$, we have that $A \notin \varepsilon C(\{A, C\}) = C(X - \{B\})$, which infers that $\{B, C\} \to A$ is not minimal. This example illustrates that it suffices to test functional dependencies $X - \{A\} \to A$, where $A\varepsilon X$ and $A\varepsilon C(X - \{B\})$ for all $B\varepsilon X$.

6. Pruning the Set Containment Lattice

If $X \to A$, then $Y \to A$ is not minimal for any proper subset Y of X. Similarly, if $Y \to A$ does not hold, then neither does $Xrightarrow A$ for any $X \subseteq Y$. The levelwise approach will only employ the former rule for discovering functional dependencies due to the movement from small to large partition sizes. Additionally, if $C(X) = \emptyset$, then $C(Y) = \emptyset$ for all supersets Y of X, and no dependency of the form $Y - \{A\} \to A$ can be minimal.

THEOREM 5 *Let $B\varepsilon X$ and $X-\{B\} \to B$. If $X \to A$, then $X-\{B\} \to A$. If X is a superkey, then $X - \{B\}$ is a superkey.*

proof
Let $X \to A$. Then $t, u\varepsilon R$ such that $t[X] = u[X] \Rightarrow t[A] = u[A]$. But $X-\{B\} \subseteq X$, thus $(t[X-\{B\}] = u[X-\{B\}] \Rightarrow t[A] = u[A])$ or $X-\{B\} \to A$. Let X be a superkey. Then Π_X is a set of singleton equivalence classes. $X-\{B\} \to B$ if and only if $|\Pi_{X-\{B\}}| = |\Pi_{(X-\{B\})\bigcup\{B\}=X}|$. Therefore the cardinality of $\Pi_{X-\{B\}}$ is the same as the cardinality of Π_X. Thus $X-\{B\}$ is a collection of singleton sets. Therefore $X - \{B\}$ is a superkey. □

The first implication in Theorem 5 allows one to remove additional attributes from the rhs candidate sets. Assume that $X - \{B\} \to B$ for some $B\varepsilon X$. Then a dependency with X on the left-hand side cannot be minimal because we can remove B from the left-hand side without changing the validity of the dependency. As a result, we can safely remove from $C(X)$ the following set:

$C''(X) = \{A\varepsilon R - X|$ there exists $B\varepsilon X$ such that $X - \{B\} \to B\}$
Furthermore, assume that X has a proper subset Y such that $Y-\{B\} \to B$ for some $B\varepsilon Y$. Then we can also remove from $C(X)$ all $A\varepsilon X - Y$. The following set of attributes can be removed from $C(X)$:

$c'''(X) = \{A\varepsilon$ X : there exists B $\varepsilon X - \{A\}$ such that $X - \{A, B\} \to B\}$. The closure of the rhs candidates, $C^+(X)$, for a set $X \subseteq R$ is defined as: $C^+(X) = (((R - C'(X)) - C''(X)) - C'''(X))$.

THEOREM 6 $C^+(X) = \{A\varepsilon R|$ *for all* $B\varepsilon X, X - \{A, B\} \to$ *B does not hold*$\}$.

Proof: $C^+(X)$

$= (((R - C'(X)) - C''(X)) - C'''(X))$

$= R - \{A\varepsilon X | X - \{A\} \to A\} - \{A\varepsilon R - X|$ there exists $B\varepsilon X$ such that $X - \{B\} \to B\} - \{A\varepsilon X|$ there exists $B\varepsilon X - A$ such that $X - \{A, B\} \to B\}$

$= R - \{A\varepsilon X|$ there exists $B\varepsilon X - A$ such that $X - \{A, B\} \to B\} - \{A\varepsilon R - X|$ there exists $B\varepsilon X$ such that$X - \{A, B\} \to B\}$

$= R - \{A\varepsilon R|$ there exists $B\varepsilon X$ such that $X - \{A, B\} \to B\}$

$= \{A\varepsilon R|$ for all $B\varepsilon X, X - \{A, B\} \to$ does not hold$\}$

□

This theorem shows that one can use the closure of the rhs candidates to test the minimality of a dependency.

THEOREM 7 *Let* $A\varepsilon X$ *and* $X - \{A\} \to A$. *The functional dependency* $X - \{A\} \to A$ *is minimal if and only if for all* $B\varepsilon X$, $A\varepsilon C^+(X - \{B\})$.

Proof: Suppose that $X - \{A\}\varepsilon A$ is not minimal, then there exists $B\varepsilon X - \{A\}$ for which $X - \{A, B\} \to A$. Then $A \notin C^+(X - \{B\})$. Therefore, for all B, if $A\varepsilon C^+(X - \{B\})$, then $X - \{A\} \to A$ is minimal. Assume that there exists $B\varepsilon X$ with $A \notin C^+(X - \{B\})$. Then there exists $C\varepsilon X - \{B\})$ such that $X - \{A, B, C\} \to C$. If $C = A$, then $B \neq A$ and $X - \{A, B\} \to A$. If $C \neq A$, then $X - \{A, B, C\} \to C$ and, consequently, $X - \{A, C\} \to C$. Thus $X - \{A, C\} \to A$ by Theorem 5. Thus $X - \{A\} \to A$ is not minimal. Therefore, if $X - \{A\} \to A$ is minimal, then, for all B, $A\varepsilon C^+(X - \{B\})$.

□

Theorem 7 gives the closure of rhs candidates two advantages over rhs candidates. First, once such a B is encountered checking can be stopped. Second, for some B, $C^+(X\{B\})$ can be empty when $C(X - \{B\})$ is not empty. The last statement implies that with the closure of rhs candidates, the set X is never even generated due to pruning.

When a key is found, additional pruning methods can be applied. $X \to A$, $A \notin X$, is tested when $X \bigcup \{A\}$ is computed since one needs $\Pi_{X \bigcup \{A\}}$ for validity testing. If X is a superkey then $X \to A$ is always valid and we do

not need $X \bigcup \{A\}$. If a superkey X is not a primary key, then a dependency $X \rightarrow A$ is not minimal for any $A \notin X$. Also, if $A \varepsilon X$ and $X - \{A\} \rightarrow A$, then by the second part of Theorem 5, $X - \{A\}$ is a superkey and we do not need Π_X for testing the validity of $X - \{A\} \rightarrow A$. X and Π_X are not required for finding the non-trivial dependencies. All keys can be deleted and thereby all their supersets can be pruned, i.e., the superkeys that are not keys.

7. Conclusion

The foundation for a new algorithmic approach for the discovery of functional and approximate functional dependencies from relations has been provided. The approach is based on partitions of row identification numbers from the relation and determining non-trivial minimal dependencies from the partitions. A breadth-first or levelwise search for the dependencies is conducted. Additionally, the search space can be pruned effectively. Both the partitions and dependencies can be computed efficiently.

References

1. Rakesh Agrawal, Heikki Mannila, Ramakrishnan Srikant, Hannu Toivonen, and A. Inkeri Verkamo. Fast discovery of association rules. *Advances in Knowledge Discovery and Data Mining* ,Usama M. Fayyad, Gregory Piatetsky–Shapiro, Padhraic Smyth, and Ramasamy Uthurusamy – editors, pages 307-328, AAAI Press, Menlo Park, CA, 1996.
2. S. Bell and P. Brockhausen. Discovery of data dependencies in relational databases. Tech. Rep. LS-8 Report-14, University of Dortmund, April 1995.
3. D. Bitton, J. Millman, and S. Torgersen. A feasibility and performance study of dependency inference. In *Proceedings of the Fifth International Conference on Data Engineering* , pages 635-641. IEEE Computer Society Press, 1989.
4. Jyrki Kivinen and Heikki Mannila. Approximate dependency inference from relations. *Theoretical Computer Science* , 149(1):129–149, 1995.
5. H. Mannila and K. J. Räihä. *The Design of Relational Databases.* Addison–Wesley, Menlo Park, California, 1992.
6. H. Mannila and K. J. Räihä. On the complexity of inferring functional dependencies. *Discrete Applied Mathematics,* 40:237-243, 1992.
7. H. Mannila and K. J. Räihä. Algorithms for inferring functional dependencies. *Data and Knowledge Engineering,* 12(1):83-99, 1994.
8. Heikki Mannila and Hannu Toivonen. Levelwise search and borders of theories in knowledge discovery. *Data Mining and Knowledge Discovery.* 1(3):241-258, 1997.
9. Z. Pawlak. *Rough Sets: The Theoretical Aspects of Reasoning About Data.* Kluwer Academic Publishers, Dordrecht, 1991.
10. Savnik and P. Flach. Bottom-up induction of functional dependencies from relations. In G. Piatetsky-Shapiro, editor, *Knowledge Discovery in Databases,* papers from the 1993 AAAI Workshop (KDD'93), pages 174-185. AAAI, 1993.

11. J. C. Schlimmer. Efficiently inducing determinations: A complete and systematic search algorithm that uses optimal pruning. In G. Piatetsky-Shapiro, editor, *Proceedings of the Tenth International Conference on Machine Learning,* pages 284-290. Morgan Kaufmann, 1993.
12. W. Ziakro. The discovery, analysis and representation of data dependencies in databases. In G. Piatetsky-Shapiro and W. G. Frawley, editors, *Knowledge Discovery in Databases,* AAAI Press / MIT Press, pages 177-195, 1991.