Theoretical Computer Science 26 (1983) 149–195
North-Holland

ORDER DEPENDENCY IN THE RELATIONAL MODEL*

Seymour GINSBURG** and Richard HULL***
Computer Science Department, University of Southern California, Los Angeles, CA 90089, U.S.A.

Communicated by M. Nivat
Received May 1982

Abstract. The relational model is formally extended to include fixed orderings on attribute domains. A new constraint, called order dependency, is then introduced to incorporate semantic information involving these orderings. It is shown that this constraint can be applied to enhance the efficiency of an implemented database. The thrust of the paper is to study logical implication for order dependency. The main theoretical results consist in (i) introducing a formalism analogous to propositional calculus for analyzing order dependency, (ii) exhibiting a sound and complete set of inference rules for order dependency, and (iii) demonstrating that determining logical implication for order dependency is co-NP-complete. It is also shown that there are sets of order dependencies for which no Armstrong relations exist.

Introduction

Since its inception over a decade ago [7], the relational model has been the foundation of many theoretical investigations into database issues. The primary vehicle in these studies for incorporating semantic information into the relational model has been through dependencies [5, 8, 11, 13, 28, 29]. In virtually all theoretical research of dependencies, an attribute domain is viewed as an abstract set, i.e., its elements are incomparable and undistinguished.[1] (This property is called 'domain independence' in [28].) However, it is obvious that many naturally arising attribute domains have an associated structure. By far the most prevalent of these structures is order and, as has been observed [10, 19, 21, 24, 26], certain semantic

* Portions of Section 1 were presented at the XP2 Conference, Pennsylvania State University, June 1981, under the title "Ordered Attribute Domains in the Relational Model".

** This author was supported in part by the National Science Foundation under Grant MCS-7925004.
*** This author was supported in part by the National Science Foundation under Grant IST-8107480.

[1] A notable exception is [21], where it is shown that tableaux can be generalized to incorporate order information. This is then used to analyze the preservation of conventional dependencies (spez., functional and join) under algebraic operators such as Codd's 'θ-join'.

information is expressible in terms of this structure. It is therefore natural to examine, from a theoretical point of view, dependencies which incorporate information involving order. The purpose of the present paper is to formally define such a dependency (called order dependency) and initiate its formal investigation, focusing primarily on logical implication.

Briefly, order dependencies permit us to formally express constraints describing the following situation: If two tuples are such that the corresponding values for certain given attributes are in a specified relationship (with respect to the domain orders), then the corresponding values for certain other given attributes are also in a specified relationship. For example, an order dependency can describe the statement that if two packages are to be shipped the same distance, then sending the heavier one will cost at least as much as sending the lighter one. As will be illustrated in the body of the paper, knowledge of such relationships can be used to enhance the efficiency of an implemented database.

The main theoretical results of the paper consist in (i) introducing a formalism analogous to propositional calculus for analyzing order dependency, (ii) exhibiting a sound and complete set of inference rules for order dependency, and (iii) demonstrating that determining logical implication for order dependency is co-NP-complete. We also show that there are sets of order dependencies for which no Armstrong relations exist.

Organizationally, the paper is divided into five sections. The first formally introduces order into attributes domains, defines order dependency, and exhibits numerous examples. Applications are presented demonstrating how order dependency can be used to improve database efficiency. Finally, a number of observations are made which pertain to satisfaction and logical implication.

Section 2 introduces and examines 'comparators', the major tool in our study of logical implication. For each set Γ of order dependencies a set $\mathscr{C}(\Gamma)$ of comparators is defined. The main results here are characterizations of (α) $\mathscr{C}(\Gamma)$ in terms of Γ, and (β) the elements of the closure of Γ (under logical implication) in terms of $\mathscr{C}(\Gamma)$. Using comparators, it is shown that sets of order dependencies need not have an Armstrong relation. In addition, a sufficiency condition is given for the existence of such a relation.

Section 3 deals with calculating $\mathscr{C}(\Gamma)$. Specifically, a formalism, based on a variant of the propositional calculus, is developed for analyzing comparators and logical implication. The major result here shows how to syntactically compute an 'order formula' which, in essence, lists the set of comparators for a set of order dependencies. The result is then applied to develop algorithms for calculating $\mathscr{C}(\Gamma)$ and determining logical implication.

In Section 4 a set of rules for inferring logical implication is presented and shown to be sound and complete. The final section considers the complexity of determining logical implication. This is shown to be co-NP-complete in a number of contexts. On the other hand, an interesting restricted context is presented for which logical implication is polynomial decidable.

1. Order dependency

In this section order dependency is introduced, several of its applications to database design and implementation are described, and some of its properties related to satisfaction are examined. These topics are considered in three successive subsections. Section 1.1 begins with a review of some fundamental concepts of the relational model. The basic notions of ordered attribute domain and order dependency, as well as some associated concepts, are then presented. In Section 1.2 several ways are explored in which order dependency can be applied to improve database efficiency. The concluding Section 1.3 introduces some simple definitions and observations pertaining to satisfaction and logical implication.

1.1. Definition of order dependency

The basis of the relational model consists in specifying a set of attributes together with their associated domains of possible values.

Definition. A *universal attribute specification* is a pair $(U_\infty, \langle \mathrm{Dom}(A) \mid A \text{ in } U_\infty \rangle)$, where (i) U_∞ is an infinite set of abstract objects (called *attributes*), and (ii) for each attribute A in U_∞, $\mathrm{Dom}(A)$ (called the *domain of* A) is an abstract set of at least two elements.[2]

We shall throughout assume some fixed universal attribute specification.

The atomic objects of the relational model are 'tuples', these corresponding to our intuitive notion of records.

Definition. A *tuple* is a function u from a finite subset U of U_∞ such that $u(A)$ is in $\mathrm{Dom}(A)$ for each A in U. For each such U, $\mathrm{Tup}(U)$ denotes the set of all tuples over U.

Tuples are combined to create 'instances', the formalization of the notion of a (flat) file.

Definition. An *instance* is a pair (U, I), or I when U is understood, where U is a finite set of attributes and I is a finite subset of $\mathrm{Tup}(U)$.

The partial orderings on attribute domains are formally incorporated by the following axiom.

[2] The assumption that each domain have at least two elements is certainly reasonable, and is included for technical reasons. E.g., it allows a simple classification in this section of the types of order of attribute domains, and it reduces the complexity of the inference rules presented in Section 4.

Axiom of Order. For each attribute A there is a partial ordering[3] $\leq_A$ on Dom(A).

It is useful to distinguish the following three types of partial orders.[4]

Definition. Let A be an attribute. Then A has *total order* if, for each pair a, a' in Dom(A), $a \leq_A a'$ or $a' \leq_A a$. A has *empty order* if, for each a, a' in Dom(A), $a \leq_A a'$ implies $a = a'$. And A has *general-partial order* if A satisfies neither of the above, i.e., if there exist a, a', b, b' in Dom(A) such that $a <_A a'$, $b \not\leq_A b'$ and $b' \not\leq_A b$.

Since each attribute domain has at least two elements, it is easily seen that the categories of total, empty and general-partial orders are exhaustive and mutually exclusive. Also, each attribute with a two-element domain has either total or empty order.

We now present examples of the three types of orders. Many attributes (e.g., SALARY, AGE, QUANTITY and DISTANCE) have as their domains a subset of the integers ordered in the usual manner, and thus have total order. Another important class of attributes having total order consists of those attributes involving chronology (e.g., DATE or TIME-OF-DAY). Also, the set of words ordered lexicographically and the collection of sets of five cards ordered according to poker rules are both totally ordered.

Examples of domains with empty order include the set of human beings, the set of phone numbers, and the set of license-plate numbers. These illustrate an important point: For our purposes, we are concerned with the meaning of attribute domains as determined by database users, not by computer (or other) representations. Thus, we may view the set of human beings as having empty order, even though computer implementations will certainly take advantage of the ordering of names in certain cases.

Finally, examples of attribute domains with general-partial order include letter grades (including P for 'pass', F for 'fail' and W for 'withdrew') with the obvious order, the set of oriented rectangles ordered by 'fits in' (as might arise in integrated circuit layout), and the set of ordered pairs of integers, where $(a, b) \leq (c, d)$ iff $a \leq c$ and $b \leq d$.

A given domain may, in different situations, be viewed as having different types of order. Thus, in some contexts it is useful to regard the set of human beings as ordered by descendency (a general-partial order) rather than by the empty order as indicated above. However, in the formal treatment here we assume each attribute domain to have a single fixed ordering (as stated by the Axiom of Order).

[3] A *partial ordering* on a set S is a binary predicate R on S such that for each a, b and c in S, (i) $R(a, a)$, (ii) if $R(a, b)$ and $R(b, c)$, then $R(a, c)$, and (iii) if $R(a, b)$ and $R(b, a)$, then $a = b$. We usually write $R(a, b)$ as $a \leq b$ or $b \geq a$. Also, we write $a < b$ or $b > a$ if $R(a, b)$ and $a \neq b$.

[4] As noted in [18], certain aspects of order can be modelled using generalized dependency constraints. Since these constraints are (by definition) equivalent to Horn clauses, however, certain other aspects of order (e.g., total order) cannot be modelled within that framework.

We now introduce notation which allows us to describe the relationship between the attribute values of pairs of tuples. To begin, we present the following.

Definition. For each attribute A, the *marked attributes* of A are the formal symbols A, $\bar{A}$, $\tilde{A}$, $\underline{A}$, $\underset{\sim}{A}$ and $\not{A}$.

Marked attributes are used in the following manner.

Notation. Let U be a finite set of attributes, A in U, u, v in Tup(U) and M a set of marked attributes of U. Write (a) $u[A]v$ if $u(A) = v(A)$, (b) $u[\bar{A}]v$ if $u(A) \leq_A v(A)$, (c) $u[\tilde{A}]v$ if $u(A) <_A v(A)$, (d) $u[\underline{A}]v$ if $u(A) \geq_A v(A)$, (e) $u[\underset{\sim}{A}]v$ if $u(A) >_A v(A)$, (f) $u[\not{A}]v$ if $u(A)$ and $v(A)$ are incomparable under $\leq_A$, and (g) $u[M]v$ if $u[\hat{B}]v$ for each marked attribute $\hat{B}$ in M.

If $X \subseteq U$ is viewed as a set of marked attributes, then $u[X]v$ if and only if[5] $u[X] = v[X]$.

Note that the structure of some of the attribute domains may render some marked attributes meaningless (e.g., $\not{A}$ can never arise if A has total order). This type of situation will be examined in more depth in Section 1.3.

We are now ready to define the central notion of order dependency. The reader will observe that functional dependency can be regarded as a special case of order dependency.

Definition. Let U be a finite set of attributes. An *order dependency* (*over* U) is an expression of the form $M \rightarrow N$, where M and N are sets of marked attributes of U such that for each A in U, $\not{A}$ is not in[6] MN. An instance (U, I) *satisfies* $M \rightarrow N$, denoted $I \models M \rightarrow N$ if, for all tuples u and v in I, $u[M]v$ implies $u[N]v$. (U, I) *satisfies* a set Γ of order dependencies, denoted $I \models \Gamma$, if $I \models \gamma$ for each γ in Γ. Finally, write $I \not\models \Gamma$ if $I \models \Gamma$ is false.

It is clear that if $I \models \Gamma$ and $J \subseteq I$, then $J \models \Gamma$.

Marked attributes of the form $\not{A}$ are prohibited in order dependency because, in analogy with functional dependency, our interest is in cases where domain values are equal or comparable. As we shall see, order dependencies as defined above can easily be applied to enhance database efficiency. However, the authors have been unable to generalize these applications to order dependencies incorporating marked attributes of the form $\not{A}$. Thus, while many of the results in the paper (a notable exception being Section 4) continue to hold if marked attributes of the form $\not{A}$ are permitted (with a modest increase in complexity), our focus is on order dependency as defined above.

[5] Let $X \subseteq U$. For each tuple u in Tup(U), $u[X]$ denotes the tuple in Tup(X) defined by $u[X](A) = u(A)$ for each A in X.

[6] As usual, XY denotes $X \cup Y$ for sets X and Y of attributes. We extend this convention to marked attributes. Also, we adopt the convention of using an attribute A to denote $\{A\}$, and extend this to marked attributes.

We now present some examples illustrating how order dependency arises in a natural manner. The first is simplistic, but is treated formally in order to illustrate below several applications of order dependency to database efficiency. Other examples are given informally.

Example 1.1. Checking-account database. Consider the information concerning the checks written (on a particular bank) by a specific individual. Let $U = \{\text{CHECK}\#, \text{DATE}, \text{PAYEE}, \text{AMOUNT}\}$ be a set of attributes. The domain of CHECK# is the set of natural numbers and represents the number of a check. The domain of DATE is the set of dates and represents the date the check was written. The domain of PAYEE is the set of names of individuals and corporations, representing to whom the check was paid. And the domain of AMOUNT is the set of dollar amounts (e.g., 24.50) and represents the amount for which the check was written. The order associated with each of the attribute domains is the obvious. (That is, for CHECK# and AMOUNT it is the restriction of the order for the real numbers, for DATE the chronological order, and for PAYEE the empty order.) At any time, the contents of the database is the instance, call it CHECKS, over the attribute set U which corresponds to the set of all checks written up to that time.

For our purposes, we assume that each check is assigned a unique number. Hence the instance CHECKS satisfies the key dependency[7] $\text{CHECK}\# \rightarrow U$. We also assume that the checks have been written in sequence in ascending order. Therefore CHECKS also satisfies the order dependencies $\overrightarrow{\text{CHECK}\#} \rightarrow \overrightarrow{\text{DATE}}$ and $\overrightarrow{\text{DATE}} \rightarrow \overrightarrow{\text{CHECK}\#}$.

Example 1.1 is representative of a wide variety of order dependencies consisting of a numerical key assigned in increasing order to transactions occurring in time. Other examples of this type include sales receipt numbers vs. time and date of purchase; serial number (of machinery) vs. date manufactured; and patent or trademark number vs. date issued. Another source of examples is when 'totals' are maintained, e.g., date vs. total production for a manufacturing plant; or date vs. odometer reading on a car or truck; or in a given year, date vs. total salary earned to date.

There are numerous examples of order dependency which do not involve time. For instance, the state sales tax increases along with the cost of a car, and the cost of units of merchandise typically falls as the total units purchased increases.

Finally, we note that order dependencies with two or more marked attributes on the left hand side arise naturally. For example, using the notation of Example 1.1 suppose $U' = \{\text{CLIENT}\} \cup U$, where Dom(CLIENT) is the set (with empty order) of all checking-account customers of a given (branch of a) bank. Then information concerning the checking account of all these customers can be stored in an instance over U'. Although such an instance would not necessarily satisfy $\overrightarrow{\text{CHECK}\#} \rightarrow \text{DATE}$, it would satisfy

$$\{\text{CLIENT}, \overrightarrow{\text{CHECK}\#}\} \rightarrow \text{DATE} \quad \text{and} \quad \{\text{CLIENT}, \overrightarrow{\text{DATE}}\} \rightarrow \overrightarrow{\text{CHECK}\#}.$$

[7] A *key dependency* is a functional dependency $X \rightarrow U$, where U is the set of all relevant attributes.

Another, more complicated, example concerns the grades assigned to students in a first- or second-year college science course (such as physics, chemistry, or biology). Typically, the final letter grade of a student is determined by adding the numerical scores given on various assignments and tests, and viewing the resulting sum relative to the corresponding sums of all other students in the class. Suppose there is a lab project (with score LAB), a midterm (with score MID) and a final exam (with score FINEX). Let the letter grade (denoted by GRADE) range over $\{A, B, C, D, F\}$. An instance holding grade information for such a class will satisfy $\{\overline{\text{LAB}}, \overline{\text{MID}}, \overline{\text{FINEX}}\} \rightarrow \overline{\text{GRADE}}$. (This obviously can be generalized to include more assignments and exams, or to include grades such as A^+ or C^+.)

1.2. Applications of order dependency

In this subsection we explore some of the ways the presence of order dependency can be applied to enhance database implementations. We begin by describing four such specific ways. The first two are of particular interest since they address the *physical* design of a relational database, whereas the applications of most other dependencies address the *logical* design of a database (for example, see [2, 5, 8, 11]). We conclude the subsection by discussing the more general problem of utilizing the presence of many order dependencies to obtain a 'good' physical design of a large relational database.

The first application of order dependency concerns reducing indexing space with no loss in retrieval time. We introduce the basic idea involved by a straightforward example using the checking-account database of Example 1.1. We then discuss three generalizations of the basic idea in order to show the wide variety of contexts in which it can be applied.

For this discussion we assume a simple model of database implementation, one in which space is measured in numbers of 'blocks' (of external storage) and response time in numbers of block 'retrievals' (into central memory) (for instance, see [27]).

Suppose now that in the checking-account database, fast access via CHECK#-value and via DATE-value is desired. A natural solution to this problem is to store the data sorted by one of these fields, implement a sparse[8] index into the values of that field, and implement a dense index into the values of the other field. It is reasonable to store the data sorted by CHECK#-value, since checks are generally written with increasing check-numbers. The order dependency $\overline{\text{CHECK\#}} \rightarrow \overline{\text{DATE}}$ can be applied as follows. Since the data satisfies the dependency and is sorted by CHECK#-value, it is also sorted by DATE-value. Thus a sparse index, instead of a dense one, can be used on DATE-values. This yields substantial savings on indexing space, as well as improves retrieval time. (In particular, if fifty records are in each block of stored data, then the indexing space is reduced by more than 96% and the retrieval time by one block retrieval.)

[8] For instance, a *sparse* index on CHECK# would list only the check number and location of the first entry of each block of stored data. A *dense* index would contain this information for essentially all entries of the stored data.

The first generalization of the above application involves cases in which an order dependency is satisfied in some approximate fashion. As shown in [9] the approach suggested here can be applied successfully in such situations to realize the same indexing space reduction without increasing retrieval time.

The second generalization involves the situation where the data cannot be stored in order of increasing CHECK#-values (for instance, because many insertions or deletions are expected, or because the data is to be sorted by some other field). Without using the order dependency $\overline{\text{CHECK\#}} \rightarrow \overline{\text{DATE}}$, two dense indexes are required, one for CHECK# and one for DATE. Knowledge of the dependency, however, allows the use of a 'combined' index for both of the fields. Such an index involves roughly 75% of the space of two dense indexes. Such a combined index can also be used for situations involving an order dependency $\bar{A} \rightarrow \bar{B}$, where A is not a key. Observe also that the multidimensional B-trees of Bentley [6] can be used efficiently in these kinds of situations.

Finally, the third generalization involves order dependencies which have more than one marked attribute on the left-hand side. These also can be exploited to enhance indexing in a database. Intuitively, such order dependencies can be used to enhance indexing in a database because (i) indexing is frequently based on sorted organization of data, and (ii) order dependency allows certain order properties to imply other order properties. To illustrate, consider an order dependency of the form $\overline{AB} \rightarrow \bar{C}$. Suppose that fast access is desired for pairs of A- and B-values, and also for C-values. A standard organization for data in this case is to arrange the data sorted primarily by A-value and secondarily by B-value. With such an organization, entries with a given value (a_0, b_0) in their A and B fields are 'clustered' in the sense that they occupy a sequence of consecutive positions in the file. As a result, sparse indexing is possible and accessing all entries with value (a_0, b_0) typically requires the retrieval of only one block of data. On the other hand, retrieving all entries with a given C-value may require an arbitrarily large number of data block retrievals. The dependency $\overline{AB} \rightarrow \bar{C}$ can be applied as follows. Roughly speaking, an increasing sequence $a_0, a_1, \ldots, a_n$ of A-values and an increasing sequence $b_0, b_1, \ldots, b_m$ of B-values can be chosen so that for each i, $1 \leq i \leq n$, and j, $1 \leq j \leq m$, the number of data records with (A, B)-pair values (a, b) satisfying $a_{i-1} \leq a \leq a_i$ and $b_{j-1} \leq b \leq b_j$ is approximately equal to the number of data records that can fit into a block of storage. (This type of data organization is explored more fully in [22].) Under this organization, entries with a fixed (A, B)-pair value are again 'clustered', so sparse indexing can be used and data block retrievals minimized. Furthermore, the dependency implies that for a given C-value c_0, the set of all entries having value c_0 generally occurs in fewer than $n+m$ of the data blocks. (In particular, if the data blocks are viewed as lying in the plane, with A-values forming one axis and B-values the other, then this set of entries typically lies within a sequence of blocks which essentially forms a diagonal line from the lower left to the upper right.) Thus, the number of data block retrievals required to obtain these records is reduced from as much as nm to roughly $n+m$ or less.

We now turn to the second application of order dependency. This involves a reduction of transmission time in a distributed database. We again present an example of the reduction in terms of the checking-account database of Example 1.1 (although quite artificial in this case). Suppose that database is implemented so that the projection, call it DATELESS, of CHECKS onto {CHECK#, PAYEE, AMOUNT} is at node N_1 and the projection, call it DATED, of CHECKS onto {CHECK#, DATE} is at node N_2. Suppose further that a user at node N_1 desires the dates of all checks with number between 100 and 200. Without knowledge of the order dependency $\overline{\text{CHECK\#}} \rightarrow \overline{\text{DATE}}$, essentially 100 dates have to be transmitted from N_2 to N_1. Using the dependency, however, a savings may be possible. Specifically, instead of sending all of the dates from N_2 to N_1, only the check number of the first check written on each day need be sent. (The date of each check is easily inferred by using the order dependency.) If on average more than one check was written for each day, then a corresponding reduction in transmission cost is realized. On the other hand, if at most one check was written per day, then the original data transmission scheme can be used. Thus, the order dependency can be applied to reduce transmission costs in certain (frequently arising) cases, without increasing the transmission cost in any case.

The above approach can be generalized to yield a resource trade-off between transmission cost and accuracy. For instance, suppose that only the approximate dates of checks numbered between 100 and 200 are desired, say with an accuracy of plus or minus three days. In that case, one need only transmit the number of the first check written in each seven-day period. (This is because the order dependency can then be used to infer the date of all checks within an accuracy of three days.) As above, the actual savings in transmission costs depend on the underlying data set.

The third application of order dependency concerns query optimization. For this we recall the work of Klug [21], which shows how orderings on attribute domains can be incorporated into tableaux. (Tableaux were originally introduced without orderings [30].) It is clear that the presence of order dependencies can be used to 'reduce' or 'simplify', in certain ways, database queries expressed with such tableaux. The reductions yield queries which take less time to answer than the original ones. (The presence of functional dependencies has already been exploited in this manner [30].)

The fourth application of order dependency concerns its use as an integrity constraint. This corresponds to the analogous application of virtually all other dependencies in the literature. Specifically, suppose that the underlying semantics of a database implies the presence of certain order dependencies, and an update is attempted which violates one of those dependencies. Then the system can be constructed to automatically determine that an error has occurred, and to perform an appropriate error recovery routine.

We conclude this subsection by considering how these various applications of order dependency might realistically be incorporated into a semi-automatic or

automatic physical design methodology.[9] In this discussion we focus primarily on the use of order dependency, although a variety of other dependencies would be incorporated. To begin, a user group or database designer would specify the data sets and relationships to be included in the database, the types and frequencies of updates and accesses anticipated, and the order (and other) dependencies which hold (either exactly or approximately). Other useful dependencies logically implied by the originally specified set can then be determined automatically (using the results of Sections 3 and 4 below). Next, a variety of potential physical designs, some of which employ the order dependency information, can be specified. These can be compared with each other, using the given update and access information to determine their relative costs. The optimal design among these can then be selected and implemented.

It is clear that the above is the skeleton of a physical design methodology, and it can be expanded and improved in many ways. Thus the methodology can be viewed as a starting point for a concerted investigation into order dependency and its use in the physical design arena. As mentioned earlier, in this paper we focus primarily on the issue of inferring logical implication among order dependencies.

1.3. *Satisfaction properties*

In this subsection we return to the formal discussion of order dependency, concentrating on simple properties related to the notion of satisfaction, i.e., of an instance satisfying a set of order dependencies. We first show that, intuitively speaking, satisfaction is 'independent' of the underlying set of attributes. We then consider the 'consistency' of sets of marked attributes and sets of order dependencies. We conclude by reviewing logical implication in the framework of order dependency.

For the result on 'independence' we use the following.

Definition. For each set M of marked attributes, the *support* of M, denoted supp(M), is the set $\{A \mid$ some marked attribute of A is in $M\}$.

Lemma 1.2. *Let (U, I) be an instance and $M \to N$ an order dependency such that* supp(MN) $\subseteq V \subseteq U$. *Then $I \models M \to N$ iff*[10] $\Pi_V(I) \models M \to N$.

The lemma obviously extends to sets Γ of order dependencies.

We turn now to the 'consistency' of sets of marked attributes and sets of order dependencies. First, recall that the structure of some attribute domains may render some marked attributes meaningless. To analyze this situation more completely, we introduce the following.

[9] The authors are indebted to Barry Jacobs for suggesting the general approach here.

[10] Let $V \subseteq U$. Then, for each instance (U, I), $\Pi_V(I)$ denotes $\{u[V] \mid u \text{ in } I\}$.

Definition. A marked attribute $\hat{A}$ is *consistent* if $u[\hat{A}]v$ for some pair u, v of tuples, and is *inconsistent* otherwise. A set M of marked attributes is *consistent* if $u[M]v$ for some pair u, v of tuples, and is *inconsistent* otherwise.

It is easily verified that if A has total order, then $\underset{\sim}{A}$ is inconsistent and all other marked attributes of A are consistent; if A has empty order, then only $\tilde{A}$ and $\underset{\sim}{A}$ are inconsistent; and if A has general-partial order, then all marked attributes of A are consistent. Also, as is readily seen, if the kind of order (i.e., empty, total or general-partial) of each relevant attribute is known, then it is decidable (in linear time) whether a given set of marked attributes is consistent.

We now turn to sets of order dependencies. It is easily verified that the empty instance (over U) satisfies each order dependency (over U). Hence, technically speaking, each set of order dependencies is consistent. However, there are order dependencies which are satisfied only by the empty instance. For example, suppose that $U = \{A, B\}$, where A and B have total order. Then for each tuple u in $\mathrm{Tup}(U)$, $u[\bar{A}]u$ but not $u[\tilde{B}]u$. Thus, no nonempty instance over U satisfies $\bar{A} \rightarrow \tilde{B}$. The preceding suggests the following, which captures the spirit of consistency for sets of order dependencies.

Definition. A set Γ of order dependencies is *proper* if some nonempty instance satisfies Γ, and is *improper* otherwise.

A finite set Γ of order dependencies is proper if no marked attribute of the form $\tilde{A}$ or $\underset{\sim}{A}$ appears on the right-hand side of any order dependency in Γ (but this condition is not necessary).

It is clear that the only sets of order dependencies which correctly arise in practice are proper. However, there are two reasons why improper sets are included in our investigation. First, a user may incorrectly specify an improper set of order dependencies, whence the need for an algorithm for checking whether a set Γ is proper or improper (see Proposition 1.3 below). And second, even if a stated result is predicated on the properness of a set of order dependencies, it is often useful to determine where this assumption was utilized, and whether or not the result (or a weakened version thereof) still holds without it (see Theorem 4.2 below).

The above definition was made without reference to an underlying set U of attributes. Using (the extended version of) Lemma 1.2, we now have the following (the proof is omitted).

Proposition 1.3. *Let Γ be a finite set of order dependencies and $U = \bigcup_{M \rightarrow N \text{ in } \Gamma} \mathrm{supp}(MN)$. Then the following three statements are equivalent:*

(i) *Γ is proper.*

(ii) *For some (any) finite $V \supseteq U$, $\{v\} \models \Gamma$ for some (any) v in $\mathrm{Tup}(V)$.*

(iii) *$\{u\} \models \Gamma$ for some (any) u in $\mathrm{Tup}(U)$.*

Thus, it is decidable whether Γ is proper.

It is clear that each set of functional dependencies, if considered as a set of order dependencies, is proper.

We conclude the section by formally defining logical implication between order dependencies, and then making a few simple observations about it. We need the following.

Definition. An *order dependency schema* (*od-schema*) is a pair (U, Γ), where U is a finite set of attributes and Γ is a set of order dependencies over U. An od-schema is *proper* (*improper*) if Γ is proper (improper).

We now have the following definition.

Definition. Let (U, Γ) be an od-schema and δ an order dependency over U. Then Γ *logically implies* δ (*relative to* U), denoted $\Gamma \models_U \delta$ (or $\Gamma \models \delta$ when U is understood) if, for each instance (U, I), $I \models \Gamma$ implies $I \models \delta$. For each set Δ of order dependencies over U, write $\Gamma \models \Delta$ if $\Gamma \models \delta$ for each δ in Δ, and $\Gamma \not\models \Delta$ otherwise. The *logical closure of* Γ (*relative to* U), denoted Γ^{+U} (or Γ^+ when U is understood), is the set $\{\delta : \Gamma \models_U \delta\}$.

It is easily verified that logical implication is transitive among sets of order dependencies, i.e., if $\Gamma_1 \models \Gamma_2$ and $\Gamma_2 \models \Gamma_3$, then $\Gamma_1 \models \Gamma_3$.

Suppose $M \to N$ is an order dependency over U, with M an inconsistent set of marked attributes. Then every instance over U satisfies $M \to N$. Therefore $\Gamma \models M \to N$ for every set Γ of order dependencies over U, including $\Gamma = \emptyset$. At the other extreme, if Γ is an improper set of order dependencies then Γ logically implies every order dependency. (This is because the empty instance satisfies each order dependency.)

The next result indicates that logical implication for order dependency is essentially independent of the underlying set of attributes, as is the case for functional dependency [4, 12]. The proof follows from Lemma 1.2, and is omitted.

Proposition 1.4. *Let* $V \subseteq U$ *and let* $\Gamma \cup \{\delta\}$ *be a set of order dependencies over* V. *Then* $\Gamma \models_V \delta$ *iff* $\Gamma \models_U \delta$.

Proposition 1.4 permits us to introduce the following.

Notation. For finite sets Γ and Δ of order dependencies, let $\Gamma \models \Delta$ denote $\Gamma \models_U \Delta$ for some (each) finite set U of attributes such that $\Gamma \cup \Delta$ is over U.

2. Comparators

In this section we introduce the theoretical foundation for our study of logical implication. A basic concept in this foundation is that of 'comparators', an analogue

for order dependencies of the notion of agreements for functional dependencies (see [3, 16]). For each set Γ of order dependencies, we associate a set $\mathscr{C}(\Gamma)$ of comparators. We then (Theorem 2.6) characterize (α) the elements of $\mathscr{C}(\Gamma)$ in terms of Γ, and (β) the elements of Γ^+ in terms of $\mathscr{C}(\Gamma)$. The results of this section are applied here to study Armstrong relations, and used in later sections for the development of mechanisms for inferring logical implication among order dependencies.

To begin the discussion, let U be a finite set of attributes, and let u and v be in Tup(U). With regards to order dependency, the salient features of u and v are the relationships, for each A in U, of $u(A)$ and $v(A)$ relative to the ordering $\leqslant_A$. To formally describe these relationships, we have the following.

Definition. For each attribute A, the *attribute-comparators* (*for* A) are the marked attributes A, $\tilde{A}$, $\underset{\sim}{A}$ and $\not{A}$. Given a finite set U of attributes, a *U-comparator* is a set of attribute-comparators, containing exactly one for each attribute in U. A *comparator* is a U-comparator for some (finite) U.

Comparators arise in a natural manner from pairs of tuples (cf. the definition of agreement in [16]).

Definition. Let U be a finite set of attributes and u, v in Tup(U). The *comparator of u with v*, denoted comp(u, v), is the set $\{\hat{A} \text{ a comparator} \mid A \text{ in } U,\ u[\hat{A}]v\}$. For each instance I, let comp(I) = $\{\text{comp}(u, v) \mid u, v \text{ in } I\}$.

For each pair u, v of tuples in Tup(U), comp(u, v) is uniquely defined, comp(u, v) is a U-comparator, and $u[\text{comp}(u, v)]v$.

For each u in Tup(U), comp(u, u) = U. Thus, given an instance I over U, U is in comp(I) iff I is nonempty.

Finally, it is easy to determine whether a U-comparator K is consistent. Indeed, K is consistent iff $\{A, \not{A}\} \cap K \neq \emptyset$ for each attribute A with empty order, and $\{A, \tilde{A}, \underset{\sim}{A}\} \cap K \neq \emptyset$ for each attribute A with total order. (This follows from the definitions of empty, total, and general-partial order, and from the fact that each attribute domain has at least two elements.)

The next result substantiates the claim that comparators completely characterize the features of pairs of tuples relevant to order dependency. The straightforward proof is omitted.

Proposition 2.1. *Let U be a finite set of attributes and let u, v, u', v' in* Tup(U) *be such that* comp(u, v) = comp(u', v'). *Then, for each set M of marked attributes over U, $u[M]v$ iff $u'[M]v'$.*

We now define the set of comparators associated with a set of order dependencies.

Definition. For each od-schema (U, Γ), the *set of* $(U$-$)$ *comparators of* Γ, denoted $\mathscr{C}(U, \Gamma)$ or $\mathscr{C}(\Gamma)$ when U is understood, is the set $\{\text{comp}(u, v) | u \text{ and } v \text{ in } I \text{ for some instance } (U, I) \text{ satisfying } \Gamma\}$.

As we shall see, the set $\mathscr{C}(\Gamma)$ of comparators associated with the set Γ of order dependencies plays a role analogous to, using the notation of [16], the sets $\mathscr{C}(\Delta)$ of closed sets associated with a set Δ of functional dependencies.

Given an od-schema (U, Γ), each element of $\mathscr{C}(\Gamma)$ is clearly consistent. Furthermore, the set of comparators of the od-schema $(U, \emptyset)$ is the set of all consistent U-comparators.

The next result relates the properness of Γ with the set $\mathscr{C}(\Gamma)$. The proof is straightforward, and is omitted.

Proposition 2.2. *The following statements are equivalent for each od-schema* (U, Γ):

(a) Γ *is proper,*

(b) $\mathscr{C}(\Gamma) \neq 0$, *and*

(c) U *is in* $\mathscr{C}(\Gamma)$.

We now describe how the set $\mathscr{C}(\Gamma)$ of comparators for Γ can be used to determine whether an instance satisfies Γ. (This is analogous to [16, Lemma 2.1].)

Proposition 2.3. *Let* (U, Γ) *be an od-schema. Then for each instance* (U, I), $I \models \Gamma$ *iff* $\text{comp}(I) \subseteq \mathscr{C}(\Gamma)$.

Proof. It is immediate that $\text{comp}(I) \subseteq \mathscr{C}(\Gamma)$ if $I \models \Gamma$. For the converse, suppose that $\text{comp}(I) \subseteq \mathscr{C}(\Gamma)$, that $M \to N$ is in Γ, and that u and v are (not necessarily distinct) tuples in I. Since $\text{comp}(u, v)$ is in $\text{comp}(I) \subseteq \mathscr{C}(\Gamma)$, there exist tuples u', v' in some instance I' satisfying Γ such that $\text{comp}(u, v) = \text{comp}(u', v')$. Since $I' \models \Gamma$, $u'[M]v'$ implies $u'[N]v'$. By Proposition 2.1, $u[M]v$ implies $u[N]v$. Hence $I \models \Gamma$. □

It is easily verified from the preceding proposition that, for fixed od-schema (U, Γ) and tuples u, v in Tup U), $\{u, v\} \models \Gamma$ iff $\text{comp}(u, v)$ is in $\mathscr{C}(\Gamma)$.

We now show how $\mathscr{C}(\Gamma)$ can be used to calculate Γ^+, the logical closure of Γ. (This development further substantiates the analogy between comparators and agreements.) To do this, we shall need two auxiliary concepts and two lemmas.

The first auxiliary concept is a generalized kind of containment between sets of marked attributes.

Notation. Let M and N be sets of marked attributes. Write $M \trianglelefteq N$ if $u[N]v$ implies $u[M]v$ for each pair u, v of tuples over a set containing $\text{supp}(MN)$; and $M \not\trianglelefteq N$ otherwise.

Intuitively, if $M \sqsubseteq N$, then N is 'at least as restrictive as' M, and hence N 'contains at least as many conditions as' M. An order dependency $N \rightarrow M$ is 'universal' (i.e., each instance over a set containing supp(MN) satisfies $N \rightarrow M$) iff $M \sqsubseteq N$. In fact, this statement still holds if the definition of order dependency is generalized to permit marked attributes of the form $\not A$.

Note that the definition of $\sqsubseteq$ implicitly depends on the types of order associated with the given attributes. (For example, if A has total order, then $A \not\sqsubseteq \bar{A}$, but if A has empty order, then $A \sqsubseteq \bar{A}$.) Also, note that if N is inconsistent, then $M \sqsubseteq N$ for each M. Finally, it is easily verified for sets M, N of marked attributes that $M \subseteq N$ implies $M \sqsubseteq N$.

The first of the two lemmas gives a 'local' characterization for when $M \sqsubseteq N$. The straightforward proof is omitted.

Lemma 2.4. *Let M and N be sets of marked attributes. Then $M \sqsubseteq N$ iff one of the following holds:*

(a) *N is inconsistent, or*

(b) *M is consistent and both of the following hold:*

(i) *For each A in* supp(MN) *with empty order,*

(α) *if $\{A, \bar{A}, \underline{A}\} \cap M \neq \emptyset$, then $\{A, \bar{A}, \underline{A}\} \cap N \neq \emptyset$, and*

(β) *if $\not A$ is in M, then $\not A$ is in N.*

(ii) *For each A in* supp(MN) *with total or general-partial order,*

(α) *if A is in M, then either A is in N or both $\bar{A}$ and $\underline{A}$ are in N,*

(β) *if $\tilde{A}(\underline{A})$ is in M, then $\tilde{A}(\underline{A})$ is in N,*

(γ) *if $\bar{A}(\underline{A})$ is in M, then at least one of $\bar{A}$, $\tilde{A}$ or A ($\underline{A}$, $\underline{\tilde{A}}$ or A) is in N, and*

(δ) *(general-partial order only) if $\not A$ is in M, then $\not A$ is in N.*

In cases where the right-hand side is consistent, $\sqsubseteq$ has a close relationship with the underlying supports, as indicated by the following straightforward result (proof omitted).

Corollary. *Let M, N, P and Q be sets of marked attributes.*

(a) *If N is consistent and $M \sqsubset N$, then* supp(M) $\subseteq$ supp(N).

(b) *If NP is consistent and* supp(M) $\cap$ supp(P) $= \emptyset$, *then $M \sqsubseteq N$ iff $M \sqsubseteq NP$.*

(c) *If PQ is consistent,* supp(P) $\cap$ supp(Q) $= \emptyset$, supp(M) $\subseteq$ supp(P) *and* supp(N) $\subseteq$ supp(Q), *then $MN \sqsubseteq PQ$ iff both $M \sqsubseteq P$ and $N \sqsubseteq Q$.*

In particular, part (c) above implies that if N is consistent, then to determine whether $M \sqsubseteq N$ one can proceed 'one attribute at a time'.

The second lemma characterizes those sets M satisfying $M \sqsubseteq$ comp(u, v) for a given pair u, v of tuples.

Lemma 2.5. *Let M be a set of marked attributes over U and let u, v be in* Tup(U). *Then $M \sqsubseteq$ comp(u, v) iff $u[M]v$.*

Proof. Suppose $M \trianglerighteq \text{comp}(u, v)$. Since $u[\text{comp}(u, v)]v$, it follows that $u[M]v$. Now suppose $u[M]v$. Since $\text{comp}(u, v)$ is consistent, it follows directly from Lemma 2.4 that $M \trianglerighteq \text{comp}(u, v)$. □

Recall that a comparator K contains exactly one attribute-comparator for each attribute in $\text{supp}(K)$. Thus, if K and L are comparators, with L consistent, then $K \trianglerighteq L$ iff $K \subseteq L$. (Clearly, $K \trianglerighteq L$ implies $K \sqsupseteq L$. The opposite implication comes from Lemma 2.4.) Hence, if X and Y are (finite) sets of attributes, then $X \subseteq Y$ iff $X \trianglerighteq Y$ (where X and Y are viewed in the latter as sets of marked attributes). Now suppose that u, v are in $\text{Tup}(U)$ and K is a comparator. Then $u[K]v$ iff (by Lemma 2.5) $K \trianglerighteq \text{comp}(u, v)$ iff $K \subseteq \text{comp}(u, v)$. And if K is a U-comparator, then $K \trianglerighteq \text{comp}(u, v)$ iff $K = \text{comp}(u, v)$.

The second of the auxiliary concepts is that of 'reversal'.

Definition. The *reversal* of a marked attribute $\hat{A}$, denoted $\hat{A}^R$, is defined by $A^R = A$, $\bar{A}^R = \underline{A}$, $\tilde{A}^R = \underline{\underline{A}}$, $\underline{A}^R = \bar{A}$, $\underline{\underline{A}}^R = \tilde{A}$ and $\underline{\bar{A}}^R = \underline{\bar{A}}$. The *reversal* of a set M of marked attributes, denoted M^R, is the set $\{\hat{A}^R \mid \hat{A} \text{ in } M\}$.

If u and v are in $\text{Tup}(U)$ and M is a set of marked attributes, then $u[M]v$ iff $v[M^R]u$. Also, $\text{comp}(u, v) = [\text{comp}(v, u)]^R$, so $\text{comp}(I)$ is closed under reversal for each instance I.

Let (U, Γ) be an od-schema. Then $\mathscr{C}(\Gamma)$ is closed under reversal, $\Gamma \models \Gamma^R$ and $\Gamma^R \models \Gamma$, where $\Gamma^R = \{M^R \to N^R \mid M \to N \text{ in } \Gamma\}$.

We are now ready for Theorem 2.6, the main result of this section. For a given od-schema (U, Γ), Theorem 2.6 characterizes the elements of $\mathscr{C}(\Gamma)$ in terms of Γ, and characterizes the elements of Γ^+ in terms of $\mathscr{C}(\Gamma)$. This result will be used as the basis for (a) determining whether $\Gamma \models \sigma$ for a given order dependency σ, (b) calculating $\mathscr{C}(\Gamma)$, and (c) calculating Γ^+. Specifically, Theorem 2.6 is applied in Section 3 to develop a 'direct' method for calculating $\mathscr{C}(\Gamma)$ and Γ^+. Also, the theorem is applied in Section 4 to obtain a direct method, based on inference rules, for performing (a) above.

Theorem 2.6. *For each od-schema (U, Γ) the following hold:*

(a) *If Γ is proper, then a consistent U-comparator K is in $\mathscr{C}(\Gamma)$ iff for each dependency $M \to N$ in Γ, $M \trianglerighteq K$ implies $N \trianglerighteq K$ and $M^R \trianglerighteq K$ implies $N^R \trianglerighteq K$. If Γ is improper, then $\mathscr{C}(\Gamma) = \emptyset$.*

(b) *$\Gamma \models M \to N$ iff for each K in $\mathscr{C}(\Gamma)$, $M \trianglerighteq K$ implies $N \trianglerighteq K$. Thus, $\Gamma^+[U] = \{M \to N$ over U for each K in $\mathscr{C}(\Gamma)$, $M \trianglerighteq K$ implies $N \trianglerighteq K\}$.*

Proof. (a) If Γ is improper, then $\mathscr{C}(\Gamma) = \emptyset$ by Proposition 2.2.

Suppose Γ is proper. Let K be in $\mathscr{C}(\Gamma)$. Then K is consistent and $K = \text{comp}(u, v)$ for some pair u, v of tuples in some $I \models \Gamma$. Clearly, $\{u, v\} \models \Gamma$. Let $M \to N$ be in Γ. Assume $M \trianglerighteq K$. Then $u[M]v$ by Lemma 2.5. Hence $u[N]v$, so $N \trianglerighteq K$ by Lemma

2.5. Assume $M^R \subseteq K$. Then $u[M^R]v$, whence $v[M]u$. Thus $v[N]u$, so $u[N^R]v$ and $N^R \subseteq K$.

Now suppose that K is a consistent U-comparator and that for each $M \to N$ in Γ, $M \subseteq K$ implies $N \subseteq K$ and $M^R \subseteq K$ implies $N^R \subseteq K$. Let $I = \{u, v\}$ be an instance such that $\text{comp}(u, v) = K$. To see that K is in $\mathscr{C}(\Gamma)$ it suffices to show that $I \models \Gamma$. Let $M \to N$ be in Γ. By Proposition 1.3 (since Γ is proper), $u[M]u$ implies $u[N]u$, and $v[M]v$ implies $v[N]v$. It remains to prove that $u[M]v$ implies $u[N]v$ and $v[M]u$ implies $v[N]u$. But these results follow from Lemma 2.5 and the fact that $v[L]u$ iff $u[L^R]v$ for each set L of marked attributes.

(b) First, suppose that Γ is improper. Then Γ logically implies each order dependency $M \to N$. On the other hand, $\mathscr{C}(\Gamma) = \emptyset$ by Proposition 2.2. Hence, the condition "$M \subseteq K$ implies $N \subseteq K$ for each K in $\mathscr{C}(\Gamma)$" is vacuously satisfied for each order dependency $M \to N$.

Now suppose that Γ is proper. Furthermore, suppose that $\Gamma \models M \to N$, K is in $\mathscr{C}(\Gamma)$ and $M \subseteq K$. Then there is an instance (U, I) satisfying Γ and a pair u, v of tuples of I such that $\text{comp}(u, v) = K$. By Lemma 2.5, $u[M]v$. Since $I \models M \to N$, $u[N]v$. By Lemma 2.5 again, $N \subseteq K$. This establishes the necessity.

To see the sufficiency, suppose that $M \subseteq K$ implies $N \subseteq K$ for each K in $\mathscr{C}(\Gamma)$. Let (U, I) be an instance satisfying Γ. Assume u and v are (not necessarily distinct) tuples in I, with $u[M]v$. Clearly $K = \text{comp}(u, v)$ is in $\mathscr{C}(\Gamma)$, and $M \subseteq K$ by Lemma 2.5. By hypothesis, $N \subseteq K$. By Lemma 2.5 again, $u[N]v$. Therefore, $I \models M \to N$ as desired.

The final sentence of the theorem is obvious. □

The condition in (a) of Theorem 2.6 that $M^R \subseteq K$ implies $N^R \subseteq K$ is used in the proof. Specifically, it is applied when showing, for u, v such that $u[K]v$, that $v[M]u$ implies $v[N]u$. Indeed, the condition cannot be dropped from the characterization. (For example, suppose $U = \{A, B\}$, with both A and B having general-partial order. Let $\Gamma = \{\tilde{A} \to \tilde{B}\}$ and $K = A\tilde{B}$. Then Γ is proper by Proposition 1.3, and K is consistent. However, K is not in $\mathscr{C}(\Gamma)$ even though $\tilde{A} \subseteq A\tilde{B}$ implies $\tilde{B} \subseteq A\tilde{B}$.)

We now apply the previous results to consider the existence of Armstrong relations for sets of order dependencies. (As shown in [25], Armstrong relations are useful in the (logical) database design process.) In particular, we first show that no Armstrong relation exists for some such sets. Then we present a sufficient (but not necessary) condition on a set of order dependencies to imply the existence of an Armstrong relation.

Following Fagin [13], we have the following.

Definition. Let (U, Γ) be an od-schema. An[11] *Armstrong relation* for Γ *(relative to* U*)* is an instance (U, I) such that $I \models \Gamma$ but $I \not\models \delta$ for each order dependency δ over U not in Γ^+.

[11] Unlike our definition, Fagin [13] allows instances to have an infinite number of tuples. Thus, an Armstrong relation as defined there may be infinite.

Suppose that (U, Γ) is an improper od-schema. Then Γ^+ is the set of all dependencies over U, and the empty instance is an Armstrong relation for Γ. (In fact, it is the only Armstrong relation for Γ.) In view of this, we only consider proper sets of order dependencies in the remainder of this section.

We now present a general condition (in terms of comparators) which implies that no Armstrong relation exists for a given set of order dependencies. We then apply this condition to exhibit a set of order dependencies with no Armstrong relation.

Lemma 2.7. *Let (U, Γ) be an od-schema. Suppose that A and B are two attributes with the properties that $\Gamma \not\models \emptyset \rightarrow A$, $\Gamma \not\models \emptyset \rightarrow B$, and for each comparator K in $\mathscr{C}(\Gamma)$ either A is in K or B is in K. Then Γ has no Armstrong relation.*

Proof. Suppose (U, I) is an instance satisfying Γ. We first show that

$$\text{either} \quad I \models \emptyset \rightarrow A \quad \text{or} \quad I \models \emptyset \rightarrow B. \tag{2.1}$$

Assume (2.1) is false. Then there are (not necessarily distinct) tuples u, v, u' and v' in I such that $u(A) \neq v(A)$ and $u'(B) \neq v'(B)$. Thus A is not in $\text{comp}(u, v)$ and B is not in $\text{comp}(u', v')$. Our assumption on comparators in $\mathscr{C}(\Gamma)$ implies that $u(B) = v(B)$ and $u'(A) = v'(A)$.

Consider $\text{comp}(u, u')$. Since $I \models \Gamma$, $\text{comp}(u, u')$ is in $\mathscr{C}(\Gamma)$. By our assumption, either A or B, say A, is in $\text{comp}(u, u')$. Then $u(A) = u'(A)$. Since $u(A) = u'(A) = v'(A)$ and $u(A) \neq v(A)$, A is neither in $\text{comp}(v, u')$ nor in $\text{comp}(v, v')$. By our assumption again, B is in both $\text{comp}(v, u')$ and $\text{comp}(v, v')$. Thus $u'(B) = v(B) = v'(B)$, a contradiction. Similarly, the assumption that B is in $\text{comp}(u, u')$ leads to a contradiction of $u(A) \neq v(A)$. Therefore, (2.1) holds.

Now suppose Γ has an Armstrong relation, say I. Since $I \models \Gamma$, either $I \models \emptyset \rightarrow A$ or $I \models \emptyset \rightarrow B$ by (2.1). However, by hypothesis, neither $\emptyset \rightarrow A$ nor $\emptyset \rightarrow B$ are in Γ^+. Therefore, $I \models \delta$ for some δ not in Γ^+. Thus, I is not an Armstrong relation, a contradiction. □

We now have the following.

Proposition 2.8. *There are sets of order dependencies with no Armstrong relation.*

Proof. Let $U = \{A, B\}$, with A and B having total order, and let $\Gamma = \{\tilde{A} \rightarrow B\}$. Clearly Γ is proper. We shall show that Γ has no Armstrong relation.

It is obvious that the comparators $\tilde{A}B$ and $A\tilde{B}$ are in $\mathscr{C}(\Gamma)$. Since $\emptyset \equiv \tilde{A}B$ and $A \not\equiv \tilde{A}B$, $\Gamma \not\models \emptyset \rightarrow A$ by Theorem 2.6. Similarly, $\Gamma \not\models \emptyset \rightarrow B$ since $A\tilde{B}$ is in $\mathscr{C}(\Gamma)$. Suppose that K is a comparator in $\mathscr{C}(\Gamma)$ and A is not in K. By definition, there is a pair u, v of tuples in some instance $I \models \Gamma$ such that $u[K]v$. Since A is not in K, $u[A] \neq v[A]$. Suppose $u(A) < v(A)$. Then $u[\tilde{A}]v$. Since $I \models \Gamma$, $u[B]v$, i.e., $u(B) = v(B)$. Thus B is in K. Suppose $v(A) < u(A)$. Since $\{u, v\} \models \Gamma$, $v[B]u$ whence

$u(B) = v(B)$ and B is again in K. In either case, B is in K. Hence, A or B is in each comparator of Γ. Thus Γ satisfies the hypotheses of Lemma 2.7, whence Γ has no Armstrong relation. □

Remark. In the above proof the set $\Gamma = \{\tilde{A} \to B\}$ was shown not to have an Armstrong relation. This situation (i.e., the absence of an Armstrong relation) is not peculiar to the presence in Γ of at least one marked attribute of the form $\tilde{A}$. For example, let $U = \{A, B, C, D, E, F\}$ be a set of six attributes, each having total order, and let $\Gamma = \{\bar{A}\bar{B} \to \bar{C}, \bar{A}\underline{B} \to \bar{D}, \bar{E}\bar{C} \to \underline{A}, \underline{E}\bar{C} \to \underline{B}, \bar{F}\bar{D} \to \underline{A}, \underline{F}\bar{D} \to \bar{B}\}$. It can be shown that Γ satisfies the condition of Lemma 2.7 (for A and B), and thus has no Armstrong relation. Γ has the interesting property that each marked attribute occurring in Γ is of the form $\bar{G}$ or $\underline{G}$.

In Lemma 2.7, comparators were used to determine the nonexistence of Armstrong relations. By contrast, we now present a condition of a set $\mathscr{C}(\Gamma)$ of comparators which is sufficient (but not necessary) to imply the existence of an Armstrong relation.

Proposition 2.9. *Given a finite set U of attributes, each with infinite domain*[12] *and each with empty or total order, let Γ be a proper set of order dependencies over U. Suppose there is a comparator K in $\mathscr{C}(\Gamma)$ with the property that for each A in U, A is in K iff $\Gamma \models \emptyset \to A$. Then Γ has an Armstrong relation.*

Proof. Since each attribute domain is infinite, without loss of generality we may assume that $S = \{-n, -(n-1), \ldots, 0, \ldots, n\} \subseteq \mathrm{Dom}(A)$ for each A in U, where n is an integer to be chosen later. We may assume further that if A has total order then the restriction of $\leq_A$ to S is the usual order on the integers.

Suppose u and v are in $\mathrm{Tup}(U)$, with the domain values of u and v all in $\{-1, 0, 1\}$. If j is in $\{0, \ldots, n-1\}$, let $v + ju$ be the tuple w in $\mathrm{Tup}(U)$ defined by $w(A) = v(A) + ju(A)$ for each A in U. For each instance I with all domain values in $\{-1, 0, 1\}$, let $I + ju = \{v + ju \mid v \text{ in } I\}$.

Let θ be the tuple in $\mathrm{Tup}(U)$ defined by $\theta(A) = 0$ for each A in U. Let K be a comparator in $\mathscr{C}(\Gamma)$ such that for each A in U, A is in K iff $\Gamma \models \emptyset \to A$. It is easily seen that there is a tuple w in $\mathrm{Tup}(U)$, all of whose domain values are in $\{-1, 0, 1\}$, such that $\theta[K]w$, i.e., $\mathrm{comp}(\theta, w) = K$. Without loss of generality, we may assume that whenever A has empty order and A is in K then $w(A) = 1$. Since K is in $\mathscr{C}(\Gamma)$, so is K^R. By Proposition 2.3, $\{\theta, w\} \models \Gamma$.

Let $\sigma_1, \ldots, \sigma_l$ be an enumeration of all the order dependencies over U not in Γ^+, and choose $n > 2l$. By Theorem 2.6, for each j, $1 \leq j \leq l$, there is a comparator

[12] The assumption that all domains be infinite allows us to avoid certain combinatorial considerations and to focus on fundamental issues (cf., [3, 16, 17]).

K_j in $\mathscr{C}(\Gamma)$ such that $M_j \subseteq K_j$ and $N_j \not\subseteq K_j$ where $\sigma_j = M_j \to N_j$. For each j, let $I_j = \{u_j, v_j\}$ be an instance with the following properties:

(i) $u_j[K_j]v_j$,

(ii) the domain values of u_j and v_j are all in $\{0, 1\}$, and

(iii) for each A in U, if $u_j(A) = v_j(A)$, then $u_j(A) = 0$.

For each j, I_j clearly exists, $I_j \models \Gamma$ (since $K_j = \text{comp}(u_j, v_j)$) and $I_j \not\models \sigma_j$ (by Lemma 2.5). Also, if $\emptyset \to A$ is in Γ^+, then (iii) implies that $u_j(A) = v_j(A) = 0$ for all j.

Let $I = \bigcup_{j=1}^{l} (I_j + 2jw)$. We shall show that I is an Armstrong relation for Γ. First, suppose that σ is an order dependency not in Γ^+. Then $\sigma = \sigma_j$ for some j, $1 \leq j \leq l$. Since $I_j \not\models \sigma_j$, it readily follows that $(I_j + 2jw) \not\models \sigma_j$. Hence $I \not\models \sigma_j$, i.e., $I \not\models \sigma$ as desired.

It remains to show that $I \models \Gamma$. By Proposition 2.3 it suffices to prove that $\text{comp}(I) \subseteq \mathscr{C}(\Gamma)$. Let u and v be in I. If $u = v$, then $\text{comp}(u, v) = U$ is in $\mathscr{C}(\Gamma)$ by Proposition 2.2 (since Γ is proper). Suppose u and v are in $I_j + 2jw$ for some j. Without loss of generality (since $\mathscr{C}(\Gamma)$ is closed under reversal) we may assume that $u = u_j + 2jw$ and $v = v_j + 2jw$. It is easily seen that $\text{comp}(u, v) = \text{comp}(u_j, v_j) = K_j$, which is in $\mathscr{C}(\Gamma)$. Finally, suppose u is in $I_j + 2jw$ and v is in $I_k + 2kw$ for some $j \neq k$. Without loss of generality, we may assume that $j < k$. We now show that $\text{comp}(u, v) = \text{comp}(\theta, w) = K$, so that $\text{comp}(u, v)$ is in $\mathscr{C}(\Gamma)$.

Let A be in U. Suppose $\emptyset \to A$ is in Γ^+. Then A is in K, $w(A) = 0$ and $u_j(A) = v_j(A) = 0 = u_k(A) = v_k(A)$. Thus, $(u_j + 2jw)(A) = 0$ and $(v_j + 2jw)(A) = 0$, whence $u(A) = 0$. Similarly $v(A) = 0$, so A is in $\text{comp}(u, v)$. Now suppose that $\emptyset \to A$ is not in Γ^+. Suppose A has empty order. Then A is in K and $w(A) = 1$. It follows from (ii) that $u(A)$ is in $\{2j, 2j + 1\}$ and $v(A)$ is in $\{2k, 2k + 1\}$. Since these two sets are disjoint, $u(A) \neq v(A)$ and A is in $\text{comp}(u, v)$. Suppose A has total order. Then $\tilde{A}$ or $\tilde{\tilde{A}}$ is in K. Assume $\tilde{A}$ is in K. Then $w(A) = 1$, so $u(A)$ is in $\{2j, 2j + 1\}$ and $v(A)$ is in $\{2k, 2k + 1\}$. Since $j < k$, $2j + 1 < 2k$. Hence $u(A) < v(A)$ and $\tilde{A}$ is in $\text{comp}(u, v)$. Finally, assume $\tilde{\tilde{A}}$ is in K. Then $w(A) = -1$, $u(A)$ is in $\{-2j, -2j + 1\}$ and $v(A)$ is in $\{-2k, -2k + 1\}$. Since $-2j > -2k + 1$, it follows that $u(A) > v(A)$ and $\tilde{\tilde{A}}$ is in $\text{comp}(u, v)$. □

Remark. The above proposition can be generalized to include general-partial orders. Briefly, let $\mathscr{I} \times \mathscr{I}$ be the set of ordered pairs of integers. Write $(a, b) \leq (a', b')$ if $a = a'$ and $b \leq b'$. Define an attribute A with general-partial order to have *strongly infinite domain* if, for each $n \geq 1$, $T(n) = \{-n, \ldots, 0, \ldots, n\} \times \{-n, \ldots, 0, \ldots, n\}$ can be embedded into $\text{Dom}(A)$ in such a way that the embedding of the $\leq$ for T coincides with $\leq_A$. Then Proposition 2.9 remains true if attributes with general-partial order having strongly infinite domain are permitted. A proof of this generalization can be constructed along lines similar to that given above. (The tuple θ is defined as before, except that for each A having general-partial order (α) $\theta(A) = (0, 0)$, and (β) w and each I_j is defined with A-values chosen from $\{-1, 0, 1\} \times \{-1, 0, 1\}$. For (appropriately chosen) tuples u and v and integer j, if A is an attribute with general-partial order and $u(A) = (u_1(A), u_2(A))$ and $v(A) = (v_1(A), v_2(A))$, then $u + jv$ is defined at A to be $(u_1(A) + jv_1(A), u_2(A) + jv_2(A))$.)

We now present two corollaries, each giving a sufficient condition for a set of order dependencies to have an Armstrong relation. The first extends to the larger context of order dependencies, the well-known result that sets of functional dependencies have Armstrong relations [1]. The second corollary concerns order dependencies in which each marked attribute appearing is of the form $\bar{A}$ or $\tilde{A}$.

Corollary 2.10. *Let U be a finite set of attributes, each with infinite domain and each with either empty or total order.*[13] *Then each set Γ of functional dependencies over U has an Armstrong relation.*

Proof. Without loss of generality we may assume $\{0, 1\} \subseteq \mathrm{Dom}(A)$ for each A in U, and $0 \leqslant_A 1$ for each A with total order. Let u and v be tuples over U with domain values in $\{0, 1\}$ such that, for each A in U, $u(A) = v(A)$ iff $\Gamma \models \emptyset \rightarrow A$. It is easily verified that $\{u, v\} \models \Gamma$. Hence $\mathrm{comp}(u, v)$ is in $\mathscr{C}(\Gamma)$. Since each set of functional dependencies is proper, the conclusion follows by Proposition 2.9. □

Corollary 2.11. *Let U be a finite set of attributes, each with infinite domain and total order.*[13] *Let Γ be a proper set of order dependencies over U such that each marked attribute appearing in Γ is either of the form $\bar{A}$ or $\tilde{A}$. Then Γ has an Armstrong relation.*

Proof. We may assume that $\{0, 1\} \subseteq \mathrm{Dom}(A)$ and $0 \leqslant_A 1$ for each A in U. Let θ and $\underline{1}$ be the tuples over U defined by $\theta(A) = 0$ and $\underline{1}(A) = 1$ for each A in U. It is easily verified that $\{\theta, \underline{1}\} \models \Gamma$, whence $\mathrm{comp}(\theta, \underline{1}) = \{\tilde{A} \mid A \text{ in } U\}$ is in $\mathscr{C}(\Gamma)$. Clearly $\mathrm{comp}(\theta, \underline{1})$ satisfies the hypotheses of Proposition 2.9, so that Γ has an Armstrong relation. □

We conclude the section by showing that the condition of Proposition 2.9 is not necessary for Γ to have an Armstrong relation.

Example 2.12. Let $U = \{A, B, C\}$ be a set of three attributes, each with total order and infinite domain. For each D in U, suppose that $\{0, 1\} \subseteq \mathrm{Dom}(D)$ and $0 \leqslant_D 1$. Let

$$I = \{(1, 0, 0), (0, 1, 0), (0, 0, 1)\}$$

and

$$\Gamma = \{\sigma \text{ an order dependency over } U \mid I \models \sigma\}.$$

Obviously, I is an Armstrong relation for Γ. Also, $\Gamma \not\models \emptyset \rightarrow A$, $\Gamma \not\models \emptyset \rightarrow B$ and $\Gamma \not\models \emptyset \rightarrow C$.

We now prove that, for each K in $\mathscr{C}(\Gamma)$, either A or B or C is in K, thereby showing that the condition of Proposition 2.9 does not hold. To see this we first note that $I \models \Gamma'$ where $\Gamma' = \{\tilde{A}\underline{B} \rightarrow C, \bar{A}\bar{B} \rightarrow \underline{C}, \tilde{A}\underline{C} \rightarrow B\}$. Thus $\Gamma' \subseteq \Gamma$. Suppose K

[13] In view of the previous remark, this corollary can be extended to include attributes with general partial order and strongly infinite domain.

is in $\mathscr{C}(\Gamma)$ and neither A nor B is in K. It suffices to show that C is in K. If $\tilde{A}\tilde{B}$ is in K, then C is in K since $\tilde{A}\tilde{B} \to C$ is in Γ'. Similarly, $\underset{\sim}{A}\underset{\sim}{B}$ is in K implies that C is in K. Suppose $\tilde{A}\underset{\sim}{B}$ is in K. Then $\underset{\sim}{C} \subseteq K$ by Theorem 2.6. By Lemma 2.4 either $\underset{\sim}{C}$ or C is in K. If $\underset{\sim}{C}$ is in K, then $\tilde{A}\underset{\sim}{C}$ in K implies B is in K, a contradiction. Thus C is in K in this case. Similarly, $\underset{\sim}{A}\tilde{B}$ in K implies that C is in K. Thus, C is in K for all possible cases of the marked attributes for A and B.

3. Calculating $\mathscr{C}(\Gamma)$

In this section we develop a formalism for analyzing comparators, relationships between pairs of tuples, and logical implication for order dependency. The formalism is based on a variant of the propositional calculus, and involves 'order variables', 'order formulas' and notions of implication and equivalence. (Although similar in spirit to the work relating propositional calculus with functional and multivalued dependencies [23], the techniques used and results obtained here are quite different from those in [23].) The key result of this section (Theorem 3.11) shows, given an od-schema (U, Γ), how to syntactically compute an order formula which, in essence, lists the set of comparators of Γ. This result is applied to develop algorithms for (α) calculating $\mathscr{C}(\Gamma)$, and (β) determining whether $\Gamma \models P \to Q$. It is interesting to note that all of the results of this section continue to hold if the definition of order dependency is relaxed to permit marked attributes of the form $\underset{\sim}{A}$. Finally, portions of the formalism introduced here (specifically, Section 3.1, and the notion of 'domain descriptor' introduced in Section 3.2) will be used in Sections 4 and 5.

The section is divided into three subsections. In Section 3.1 a formalism is developed which, intuitively speaking, allows marked attributes to be viewed as propositional variables and order dependencies as propositional formulas. Notions of implication and equivalence are defined for the formulas. The formalism is used to specify an order formula which, for proper od-schemas (U, Γ), characterizes the elements of $\mathscr{C}(\Gamma)$ (Proposition 3.3).

In Section 3.2 the order formula is extended and modified so that, speaking intuitively, it is in conjunctive normal form (Proposition 3.7). Section 3.3 applies the above result. First, it is shown how to transform the conjunctive normal form into disjunctive normal form, and from this how to compute $\mathscr{C}(\Gamma)$ (Theorem 3.11). Then the two algorithms mentioned earlier, (α) and (β), are developed.

Before beginning, note that it is easy to determine from Proposition 1.3 whether Γ is improper. If so, then $\mathscr{C}(\Gamma) = \emptyset$ and Γ^+ is the set of all order dependencies over U. Thus the focus will be on the case when Γ is proper. Also, we assume that U is a nonempty finite set of attributes. Finally, as is readily seen, $\Gamma^+ = (\Gamma \cup \{A \to A\})^+$ where A is some (any) attribute in U. Thus, without loss of generality, we assume for this section that a given set Γ always contains at least one order dependency with nonempty right-hand side.

3.1. Order variables and order formulas

To start the formal discussion, let U be a fixed nonempty finite set of attributes. The marked attributes of U will be the variables in our modified propositional calculus.[14]

Definition. An (*order*) *variable* (*over* U) is a marked attribute of U.

The formulas of the system are defined recursively as follows.

Definition. The family of (*order*) *formulas* (*over* U) is the smallest family of expressions containing each order variable over U, and, for each pair ϕ_1, ϕ_2 of formulas in this family, the expressions[15] (i) $(\phi_1 \wedge \phi_2)$, (ii) $(\phi_1 \vee \phi_2)$, (iii) $(\phi_1 \rightarrow \phi_2)$, and (iv) $(\neg\phi_1)$.

Formulas are used to describe properties of U-comparators (both consistent and inconsistent) in the following way.

Definition. Let K be a U-comparator and ϕ a formula. Then K *satisfies* ϕ, denoted $K \models \phi$, if

(i) $\phi = \hat{A}$, where $\hat{A}$ is a[16] comparator variable in K,
(ii) $\phi = \tilde{A}$, where A or $\tilde{A}$ is in K,
(iii) $\phi = \underset{\sim}{A}$, where A or $\underset{\sim}{A}$ is in K,
(iv) $\phi = \phi_1 \wedge \phi_2$, $K \models \phi_1$ and $K \models \phi_2$,
(v) $\phi = \phi_1 \vee \phi_2$ and either $K \models \phi_1$ or $K \models \phi_2$,
(vi) $\phi = \phi_1 \rightarrow \phi_2$ and either $K \models \phi_1$ is false or $K \models \phi_2$ is true, and
(vii) $\phi = \neg\phi_1$ and $K \models \phi_1$ is false.

Finally, write $K \not\models \phi$ if $K \models \phi$ is false.

Note that the above definition is made with no reference, either implicitly or explicitly, to the type of order of the underlying attribute domains.

Some formulas are satisfied by all U-comparators (e.g., $\phi \rightarrow \phi$ for each formula ϕ and $((\tilde{A} \vee \underset{\sim}{A}) \vee \hat{A})$ for each attribute A). And some formulas are satisfied by no U-comparator (e.g., $\phi \wedge (\neg\phi)$ for each formula ϕ, and $(A \wedge \hat{A})$ for each attribute A).

The interpretation of the formulas given above naturally leads to the following notions of implication and equivalence.

Definition. Let ϕ_1 and ϕ_2 be formulas over U. Then ϕ_1 *implies* ϕ_2 if, for each U-comparator K, $K \models \phi_1$ implies $K \models \phi_2$. The formulas ϕ_1 and ϕ_2 are equivalent, denoted $\phi_1 \equiv \phi_2$, if ϕ_1 implies ϕ_2 and ϕ_2 implies ϕ_3.

[14] Technically, the symbols used for variables should be distinct from those used for marked attributes. However, the intended meaning of the symbols will be clear from the context.

[15] Technically speaking, the arrow '$\rightarrow$' used in order dependencies and the arrow '$\rightarrow$' used in order formulas are to be viewed as different symbols.

[16] A *comparator variable* is an order variable of the form A, $\tilde{A}$, $\underset{\sim}{A}$ or $\hat{A}$.

It is clear that implication is transitive among formulas, and that equivalence is an equivalence relation. Furthermore, conjunction ($\wedge$) and disjunction ($\vee$) are commutative and associative (i.e., $(\phi_1 \wedge \phi_2) \equiv (\phi_2 \wedge \phi_1)$ and $((\phi_1 \wedge \phi_2) \wedge \phi_3) \equiv (\phi_1 \wedge (\phi_2 \wedge \phi_3))$, and similarly for $\vee$). In view of this, we adopt the convention of omitting parentheses from formulas if the intended (equivalence class of) formula(s) is unambiguous.

There is a number of other equivalences which will be of use to us in the sequel. We now list them (without proof).

Proposition 3.1. *Let ϕ_1, ϕ_2 and ϕ_3 be formulas and Φ a nonempty set of formulas. Then*

(i) $(\phi_1 \rightarrow \phi_2) \equiv ((\neg \phi_1) \vee \phi_2)$,

(ii) $(\phi_1 \wedge (\phi_2 \vee \phi_3)) \equiv ((\phi_1 \wedge \phi_2) \vee (\phi_1 \wedge \phi_3))$,

(iii)[17] $\neg(\bigwedge \Phi) \equiv \bigvee \{\neg \phi \mid \phi \text{ in } \Phi\}$, *and*

(iv) $\phi_i \equiv \psi_i$, $1 \leq i \leq n$, *implies that* $\bigwedge\{\phi_1, \ldots, \phi_n\} \equiv \bigwedge\{\psi_1, \ldots, \psi_n\}$ *and* $\bigvee\{\phi_1, \ldots, \phi_n\} \equiv \bigvee\{\psi_1, \ldots, \psi_n\}$.

We now turn to the interpretation of order dependencies within the above framework. To motivate this interpretation, we first introduce notation which allows us to describe, using formulas, relationships between pairs of tuples.

Notation. For tuples u, v in $\mathrm{Tup}(U)$ and a formula ϕ, write $u[\phi]v$ if $\mathrm{comp}(u, v) \models \phi$.

We now have the following.

Notation. For each nonempty set M of marked attributes, the *(order) formula* of M, denoted $\phi(M)$, is the formula $\bigwedge\{\hat{A} \mid \hat{A} \text{ in } M\}$. For an order dependency $M \rightarrow N$, with $N \neq \emptyset$, the *(order) formula* of $M \rightarrow N$, denoted $\phi(M \rightarrow N)$, is the formula $\phi(M) \rightarrow \phi(N)$ if M is nonempty, and $\phi(N)$ if M is empty.

For a nonempty set M of marked atributes and tuples u, v in $\mathrm{Tup}(U)$, it is clear that $u[M]v$ iff $u[\phi(M)]v$. The following easily verified lemma (therefore the proof is omitted) formally states the relationship between an order dependency $M \rightarrow N$, $N \neq \emptyset$, and its corresponding formula $\phi(M \rightarrow N)$.

Lemma 3.2. *Let $M \rightarrow N$ be an order dependency over U with $N \neq \emptyset$, and let u and v be in $\mathrm{Tup}(U)$. Then $u[M]v$ implies $u[N]v$ iff $u[\phi(M \rightarrow N)]v$.*

We conclude the first subsection by presenting, for a proper od-schema (U, Γ), a formula which characterizes when a consistent U-comparator is in $\mathscr{C}(\Gamma)$.

[17] For a nonempty set $\Psi = \{\psi_1, \ldots, \psi_n\}$ of formulas, $\bigwedge\Psi$ denotes $\psi_1 \wedge \psi_2 \wedge \cdots \wedge \psi_n$ and $\bigvee\Psi$ denotes $\psi_1 \vee \psi_2 \vee \cdots \vee \psi_n$.

Proposition 3.3. *Let* (U, Γ) *be a proper od-schema, let*[18]

$$\Phi = \bigwedge\{\phi(M \to N), \phi(M^R \to N^R) \mid M \to N \text{ in } \Gamma, N \neq \emptyset\},$$

and let K *be a consistent* U*-comparator. Then* K *is in* $\mathscr{C}(\Gamma)$ *iff* $K \models \Phi$.

Proof. Since K is consistent, there are tuples u, v in $\mathrm{Tup}(U)$ such that $\mathrm{comp}(u, v) = K$. Suppose that K is in $\mathscr{C}(\Gamma)$. Then $\{u, v\} \models \Gamma$. Let $M \to N$ be in Γ with $N \neq \emptyset$. Then $u[M]v$ implies $u[N]v$, whence $u[\phi(M \to N)]v$ by Lemma 3.2. Also, $v[M]u$ implies $v[N]u$, so $u[M^R]v$ implies $u[N^R]v$. By Lemma 3.2, $u[\phi(M^R \to N^R)]v$. Since this holds for each order dependency $M \to N$ in Γ with $N \neq \emptyset$, it follows that $u[\Phi]v$, i.e., $K \models \Phi$.

Now suppose that $K \models \Phi$, i.e., $u[\Phi]v$. To show that $\mathrm{comp}(u, v)$ is in $\mathscr{C}(\Gamma)$, it suffices to show that $\{u, v\} \models \Gamma$. Let $M \to N$ be in Γ. By Proposition 1.3 (since Γ is proper), $u[M]u$ implies $u[N]u$, and $v[M]v$ implies $v[N]v$. Suppose $u[M]v$. If $N = \emptyset$, then $u[N]v$ trivially. If $N \neq \emptyset$, then $\phi(M \to N)$ is a conjunct of Φ, so $u[\phi(M \to N)]v$. Since $u[M]v$, $u[N]v$ by Lemma 3.2. Finally, suppose that $v[M]u$. Again, if $N = \emptyset$, then $v[N]u$ trivially. Thus suppose $N \neq \emptyset$. Then $\phi(M^R \to N^R)$ is a conjunct in Φ, whence $u[\phi(M^R \to N^R)]v$. Since $v[M]u$, it follows that $u[M^R]v$. By Lemma 3.2, $u[N^R]v$. Thus $v[N]u$. □

3.2. *Reduction to conjunctive normal form*

In this subsection we modify the formula Φ of Proposition 3.3 so that it is (essentially) in conjunctive normal form. The modified formula is then extended in such a way that it characterizes when an arbitrary U-comparator is in $\mathscr{C}(\Gamma)$ (in contrast to the formula Φ, which characterizes when a *consistent* U-comparator is in $\mathscr{C}(\Gamma)$). This development is summarized in Proposition 3.7.

As a first step, we define a particular type of normal form for order formulas and prove, in essence, that each order dependency formula has a normal form equivalent.

Definition. An order formula is in *restricted normal form* if it is of the shape $\hat{A}_1 \vee \cdots \vee \hat{A}_n$, where $n \geq 1$ and each $\hat{A}_i$ is a comparator variable.

We first show (Lemma 3.4) that for each order dependency of the shape $M \to \hat{A}$ (where $\hat{A}$ is a marked attribute), there is a formula ϕ, in restricted normal form, which is equivalent to $\phi(M \to \hat{A})$.

Definition. The *signature*, denoted σ, of variables and of negated variables is defined as follows:

(i) $\sigma(\hat{A}) = \hat{A}$ if $\hat{A}$ is a comparator variable,

(ii) $\sigma(\bar{A}) = A \vee \hat{A}$,

[18] This formula is well defined since, for this section, we assume that Γ always contains at least one order dependency with nonempty right-hand side.

(iii) $\sigma(\underset{\sim}{A}) = A \vee \underset{\sim}{A}$, and

(iv) $\sigma(\neg\hat{A}) = \bigvee \{A' | A'$ is a comparator variable for A and does not occur in $\sigma(\hat{A})\}$.

It is easily verified that for each variable $\hat{A}$, (i) $\hat{A} \equiv \sigma(\hat{A})$, and (ii) $\neg\hat{A} \equiv \sigma(\neg\hat{A})$. Furthermore, the signature of each variable and of each negated variable is a formula in restricted normal form.

The concept of signature is now extended to order dependencies with shape $M \to \hat{A}$. Note that these signatures are again in restricted normal form.

Definition. Let $M \to \hat{A}$ be an order dependency with $\hat{A}$ a marked attribute. The *signature* of $M \to \hat{A}$, denoted $\sigma(M \to \hat{A})$, is the formula[19] $(\bigvee \{\sigma(\neg\hat{B}) | \hat{B}$ in $M\}) \vee \sigma(\hat{A})$.

The motivation for this definition is given in the following lemma.

Lemma 3.4. *For each order dependency $M \to \hat{A}$, $\hat{A}$ a marked attribute, $\phi(M \to \hat{A}) \equiv \sigma(M \to \hat{A})$.*

Proof. If $M = \emptyset$, then $\phi(M \to \hat{A}) = \phi(\hat{A}) = \hat{A} \equiv \sigma(\hat{A}) = \sigma(M \to \hat{A})$. Now suppose $M \neq \emptyset$. Then

$$\begin{aligned}
\phi(M \to \hat{A}) &= \phi(M) \to \phi(\hat{A}) && \text{by definition} \\
&= \phi(M) \to \hat{A} && \text{by definition} \\
&\equiv ((\neg\phi(M)) \vee \hat{A}) && \text{by Proposition 3.1(i)} \\
&= (\neg(\bigwedge \{\hat{B} | \hat{B} \text{ in } M\})) \vee \hat{A}) && \text{by definition} \\
&\equiv (\bigvee \{\neg B | \hat{B} \text{ in } M\}) \vee \hat{A}) && \text{by Proposition 3.1(iii)} \\
&\equiv ((\bigvee \{\sigma(\neg\hat{B}) | \hat{B} \text{ in } M\}) \vee \sigma(\hat{A})) && \text{by Proposition 3.1(iv)} \\
&= \sigma(M \to \hat{A}) && \text{by definition.} \quad \square
\end{aligned}$$

The notion of signature is next extended to sets of order dependencies. Here, the signature will be a conjunction of signatures of certain dependencies, and has shape analogous to the conjunctive normal form of propositional calculus.

Definition. Let (U, Γ) be an od-schema. The *signature* of Γ, denoted $\Sigma(\Gamma)$ or Σ if Γ is understood, is the formula[20]

$$\bigwedge \{\sigma(M \to \hat{A}), \sigma(M^R \to \hat{A}^R) | \hat{A} \text{ in } N \text{ for some } M \to N \text{ in } \Gamma\}.$$

[19] If $M = \emptyset$, then $(\bigvee \{\sigma(\neg\hat{B}) | \hat{B} \text{ in } M\}) \vee \sigma(\hat{A})$ is to mean $\sigma(\hat{A})$.

[20] See Footnote 17.

The inclusion of the conjuncts $\sigma(M^R \to \hat{A}^R)$ is to allow $\Sigma(\Gamma)$ to play the role of the formula Φ of Proposition 3.3, as will now be shown.

Lemma 3.5. *Let (U, Γ) be a proper od-schema and K a consistent U-comparator. Then K is in $\mathscr{C}(\Gamma)$ iff $K \models \Sigma(\Gamma)$.*

Proof. Let $\Gamma' = \{M \to \hat{A} \mid \hat{A} \text{ in } N \text{ for some } M \to N \text{ in } \Gamma\}$. It is easily verified that, for each instance I, $I \models \Gamma$ iff $I \models \Gamma'$. Therefore Γ' is proper and $\mathscr{C}(\Gamma') = \mathscr{C}(\Gamma)$. Let

$$\Phi = \bigwedge \{\phi(M \to \hat{A}), \phi(M^R \to \hat{A}) \mid M \to \hat{A} \text{ in } \Gamma'\}.$$

By Lemma 3.4, $\Phi \equiv \Sigma(\Gamma)$. The result then follows from Proposition 3.3. □

As we have seen, the formula $\Sigma(\Gamma)$ can be regarded as a conjunction of restricted normal forms which encode the dependencies in Γ. A similar encoding is now developed for the types of order of the underlying attribute domains, i.e., empty, total, or general-partial. This is needed so that the hypothesis of K being consistent in Lemma 3.5 can be dropped.

Definition. For each attribute A in U, let $\delta(A)$, called the *domain specification* of A, be

(i) $\tilde{A} \vee A \vee \underset{\sim}{A}$ if A has total order,

(ii) $A \vee \not{A}$ if A has empty order, and

(iii) $\tilde{A} \vee A \vee \underset{\sim}{A} \vee \not{A}$ if A has general-partial order.

$\Delta(U)$ (or Δ if U is understood), called the *domain specification* of U, is the formula $\bigwedge \{\delta(A) \mid A \text{ in } U\}$.

It is easily seen that Δ is a conjunction of restricted normal forms. The next lemma substantiates the claim that Δ is a syntactic encoding of the types of order of the underlying attribute domains. (The straightforward proof is omitted.)

Lemma 3.6. *A U-comparator K is consistent iff $K \models \Delta(U)$.*

Combining Lemmas 3.5 and 3.6, we now show that the comparators of $\mathscr{C}(\Gamma)$ are characterized by a conjunction of restricted normal forms.

Proposition 3.7. *Let (U, Γ) be a proper od-schema and K a U-comparator. Then K is in $\mathscr{C}(\Gamma)$ iff $K \models \Sigma(\Gamma) \wedge \Delta(U)$.*

Proof. Let K be in $\mathscr{C}(\Gamma)$. Clearly K is consistent. Thus $K \models \Sigma(\Gamma)$ by Lemma 3.5 and $K \models \Delta(U)$ by Lemma 3.6, and so $K \models \Sigma(\Gamma) \wedge \Delta(U)$. Conversely, suppose $K \models \Sigma(\Gamma) \wedge \Delta(U)$. Thus $K \models \Delta(U)$ and $K \models \Sigma(\Gamma)$, whence K is consistent by Lemma 3.6, and K is in $\mathscr{C}(\Gamma)$ by Lemma 3.5. □

3.3. Applications

The results of the previous two subsections are now applied to obtain three useful results. The first involves transforming the formula $\Sigma(\Gamma) \wedge \Delta(U)$ into (what is essentially) disjunctive normal form, and describing how the resulting formula can be simplified to 'list' the elements of $\mathscr{C}(\Gamma)$ (Theorem 3.11). In turn, this result is used as a basis for an algorithm (Algorithm 3.12) which specifies a direct method for computing $\mathscr{C}(\Gamma)$. Finally, the algorithm is refined to obtain another algorithm (Algorithm 3.13), which specifies a procedure for determining whether $\Gamma \models P \rightarrow Q$.

To obtain the first result, we need the following.

Notation. Let (U, Γ) be a proper od-schema. Let

$$\Sigma \wedge \Delta = \bigwedge \{\hat{A}_1^i \vee \cdots \vee \hat{A}_{i_k}^i \mid 1 \leq i \leq n\},$$

where each $\hat{A}_j^i$ is a comparator variable. Let

$$\Omega = \bigvee \{\hat{A}_{j_1}^1 \wedge \cdots \wedge \hat{A}_{j_n}^n \mid 1 \leq j_i \leq i_k \text{ for each } i, i \leq 1 \leq n\}$$

$$= \bigvee \{\omega_p \mid 1 \leq p \leq q\}.$$

It is clear that Ω has form analogous to disjunctive normal form of propositional calculus, and $\Sigma \wedge \Delta \equiv \Omega$ by Proposition 3.1(ii). The following lemma (proof omitted) now easily follows from Proposition 3.7.

Lemma 3.8. *The following are equivalent for each U-comparator K:*

(i) *K is in $\mathscr{C}(\Gamma)$,*

(ii) *$K \models \Omega$, and*

(iii) *$K \models \omega_p$ for some p, $1 \leq p \leq q$.*

Intuitively, Lemma 3.8 can be interpreted as follows. Each conjunction ω_p specifies a condition on U-comparators which is sufficient to imply that K is in $\mathscr{C}(\Gamma)$. Furthermore, the set $\{\omega_p \mid 1 \leq p \leq q\}$ is 'complete' in the sense that each comparator K in $\mathscr{C}(\Gamma)$ satisfies at least one ω_p.

We now consider the possible forms of the conjunctions ω_p. First, we note that the following holds.

Lemma 3.9. *For each p, $1 \leq p \leq q$, and each attribute A, at least one variable over A occurs in ω_p.*

Proof. By definition, the conjunction ω_p involves exactly one variable from each of the disjunctions $\hat{A}_1^i \vee \cdots \vee \hat{A}_{i_k}^i$ of $\Sigma \wedge \Delta$. Since the domain specification $\delta(A)$ is one of the disjunctions, at least one variable over A occurs in ω_p. □

Second, we note that two 'conflicting' variables (e.g., A and $\tilde{A}$ for a given attribute A) can appear in ω_p. To distinguish these cases we introduce the following.

Definition. A conjunction $\bigwedge_{i=1}^{m} \hat{A}_i$ of $m \geq 1$ comparator variables $\hat{A}_i$ is *conflicting* if, for some attribute A, the set $\{A, \underset{\sim}{A}, \tilde{A}, \not{A}\} \cap \{\hat{A}_i \mid 1 \leq i \leq m\}$ has more than one element, and is *nonconflicting* otherwise.

It is clear that no U-comparator satisfies a conflicting conjunction. This suggests the following.

Notation. Let $\{\psi_s \mid 1 \leq s \leq t\}$ be the set of all nonconflicting ω_p, $1 \leq p \leq q$, and let $\Psi = \bigvee_{s=1}^{t} \psi_s$.

From Lemma 3.8 we immediately get the following lemma.

Lemma 3.10. *The following are equivalent for each U-comparator K:*

(i) *K is in $\mathscr{C}(\Gamma)$,*

(ii) *$K \models \Psi$, and*

(iii) *$K \models \psi_s$ for some s, $1 \leq s \leq t$.*

Thus, the set $\{\psi_s \mid 1 \leq s \leq t\}$ can be viewed as a 'complete' set of conditions sufficient to imply that a U-comparator is in $\mathscr{C}(\Gamma)$, the same as $\{\omega_p \mid 1 \leq p \leq q\}$.

The following provides a mechanism for considering each conjunction ψ_s as a set of attribute comparators.

Notation. For each nonconflicting conjunction $\psi = \bigwedge_{i=1}^{n} \hat{A}_i$ of comparator variables, let $K(\psi) = \{\hat{A}_i \mid 1 \leq i \leq n\}$.

For ψ as above, $K(\psi)$ is a comparator (although not necessarily a U-comparator).

We are now ready for the main result of the section.

Theorem 3.11. *Let (U, Γ) be a proper od-schema and let $K(\Psi) = \{K(\psi_s) \mid 1 \leq s \leq t\}$. Then $\mathscr{C}(\Gamma) = K(\Psi)$.*

Proof. We first show that

each element of $K(\Psi)$ is a U-comparator. (3.1)

Thus let K be in $K(\Psi)$. Then $K = K(\psi_s)$ for some s, $1 \leq s \leq t$. Since ψ_s is nonconflicting and $\psi_s = \omega_p$ for some p (so that $\text{supp}(K) = U$ by Lemma 3.9), $K(\psi_s)$ is a U-comparator, i.e., (3.1) holds.

Now suppose K is a U-comparator. By Lemma 3.10, K is in $\mathscr{C}(\Gamma)$ iff $K \models \psi_s$ for some s, $1 \leq s \leq t$. Since $K(\psi_s)$ is a U-comparator by (3.1), $K \models \psi_s$ iff $K = K(\psi_s)$. The equality $\mathscr{C}(\Gamma) = K(\Psi)$ now follows. □

As mentioned earlier, the above theorem holds if the definition of order dependency is altered to allow marked attributes of the form $\not{A}$. Also, since functional

dependency is a special case of order dependency, the development of this section can be used to gain insight into logical implication and closed sets for functional dependency. Finally, it can be shown that if (U, Γ) is an arbitrary od-schema, then Γ is proper iff U is in $K(\Psi)$.

We now summarize the development leading up to Theorem 3.11. Beginning with a proper od-schema (U, Γ), the order formulas $\Sigma(\Gamma)$ and $\Delta(U)$ were calculated (in a purely formal manner). Using a formal manipulation analogous to the transformation from conjunctive normal form to disjunctive normal form in propositional calculus, the formula $\Sigma(\Gamma) \wedge \Delta(U)$ was transformed into the formula $\Omega = \bigvee \{\omega_p \mid 1 \leq p \leq q\}$, where each ω_p is a conjunction of comparator variables. Then Ω was reduced to $\Psi = \bigvee \{\psi_s \mid 1 \leq s \leq t\}$ by removing all of the conflicting conjunctions in Ω, again a formal manipulation. And finally, the set of U-comparators $K(\Psi) = \{K(\psi_s) \mid 1 \leq s \leq t\}$ was readily computed from Ψ. Thus, the formalism introduced in this section can be mechanically used to build a formula Ψ which essentially 'lists' $\mathscr{C}(\Gamma)$.

We now present two (informally described) algorithms based on the above development. In particular, we (α) describe a direct method of calculating $\mathscr{C}(\Gamma)$ from (what is essentially) $\Sigma \wedge \Delta$, and (β) modify this method to determine whether $\Gamma \models P \to Q$. The proofs of the correctness of these algorithms are straightforward and omitted.

Algorithm 3.12. Calculating $\mathscr{C}(\Gamma)$ from (U, Γ).

Input: (U, Γ).

Output: $\mathscr{C}(\Gamma)$.

Procedure: Use Proposition 1.3 to determine if Γ is proper. If Γ is improper, then $\mathscr{C}(\Gamma) = \emptyset$. Suppose Γ is proper. Calculate

$$\mathscr{I} = \{\sigma(M \to \hat{A}), \sigma(M^R \to \hat{A}^R) \mid M \to N \text{ in } \Gamma, \hat{A} \text{ in } N\} \quad \text{and} \quad \mathscr{S} = \{\delta(A) \mid A \text{ in } U\}.$$

Using $\mathscr{S}$, list the family $\{K_1, \ldots, K_n\}$ of consistent U-comparators. (This can be done by listing, in some uniform way (e.g., based on backtracking) the family of all sets consisting of exactly one comparator variable from each formula in $\mathscr{S}$. The resulting family is the desired one since, for each A in U, $\delta(A)$ is the disjunction of all consistent comparator variables for A.) For each K_i, determine whether, for each formula $\hat{A}_1 \vee \cdots \vee \hat{A}_m$ in $\mathscr{I}$,

$$\text{there is some } j,\ 1 \leq j \leq m, \text{ such that } \hat{A}_j \text{ is in } K_i. \tag{3.2}$$

By Lemma 3.5, K_i is in $\mathscr{C}(\Gamma)$ iff (3.2) holds for each formula in $\mathscr{I}$.

The above algorithm can be modified to obtain an improvement in performance. However, since[21] $\#(\mathscr{C}(\Gamma))$ is potentially exponential in $\#(U)$, every algorithm which computes $\mathscr{C}(\Gamma)$ has, at worst, running time exponential in $\#(U)$.

[21] For each finite set F, $\#(F)$ is the number of elements in F.

Algorithm 3.13. Determining whether $\Gamma \models P \to Q$.

Input: (U, Γ) and $P \to Q$.

Output: Determining whether $\Gamma \models P \to Q$.

Procedure: As in Algorithm 3.12, use Proposition 1.3 to determine whether Γ is proper. If Γ is improper, then $\Gamma \models P \to Q$. Suppose Γ is proper. Check whether P is consistent. If P is inconsistent, then $\Gamma \models P \to Q$. Suppose that P is consistent. Calculate

$$\mathscr{S} = \sigma\{(M \to \hat{A}), \sigma(M^{R} \to \hat{A}^{R}) \mid M \to N \text{ in } \Gamma, \hat{A} \text{ in } N\}.$$

For each A in supp(P), compute

$$\delta'(A) = \bigvee \{\hat{A} \mid \hat{A} \text{ is a consistent attribute comparator over } A \text{ and } \{\hat{\hat{A}} \text{ in } P \mid \hat{\hat{A}} \text{ a marked attribute over } A\} \sqsubseteq \hat{A}\}.$$

Set

$$\mathscr{S}' = \{\delta'(A) \mid A \text{ in supp}(P)\} \cup \{\delta(A) \mid A \text{ in } U - \text{supp}(P)\}.$$

Using $\mathscr{S}'$, determine the family $\{K_1, \ldots, K_n\}$ of consistent U-comparators such that $P \sqsubseteq K_i$ for each i, $1 \leq i \leq n$. (The technique of Algorithm 3.12 is employed here, with $\mathscr{S}'$ being used instead of $\mathscr{S}$.) As in Algorithm 3.12, select from this family the subfamily $\{K_{i_1}, \ldots, K_{i_l}\}$ of comparators K such that, for each formula $\hat{A}_1 \vee \cdots \vee \hat{A}_m$ in $\mathscr{S}$, $\hat{A}_j$ is in K for some j, $1 \leq j \leq m$. For each k, $1 \leq k \leq l$, check whether $Q \sqsubseteq K_{i_k}$. If this is true for each k, then $\Gamma \models P \to Q$ (by Theorem 2.6(b)). If not, then $\Gamma \models P \to Q$ is false.

As with Algorithm 3.12, this algorithm has at worst exponential running time. Since deciding logical implication is, in the general case, co-NP-complete (see Section 5), it is likely that exponential running time cannot be substantially reduced.

4. Inference rules

Inference mechanisms based on sound and complete sets of inference rules and formal proofs have been developed for functional [1, 12], multivalued [4] and algebraic [28] dependencies. In this section we present such a mechanism for order dependencies.

We begin by briefly reviewing the well-known notions of 'inference rules', formal 'proof' using inference rules, 'soundness' and 'completeness' [4, 27] with respect to order dependencies.

An (order dependency) inference rule is a formal expression used to deduce an order dependency a set of other order dependencies (and possibly from set-theoretic relationships). Using this (admittedly informal) basis, we have the following.

Definition. Let (U, Γ) be an od-schema and $\mathcal{I}$ a set of inference rules. A *proof* from Γ using $\mathcal{I}$ (relative to U) is a sequence $\gamma_1, \ldots, \gamma_n$ of order dependencies over U such that for each i, $1 \leq i \leq n$, either γ_i is in Γ or γ_i follows from the application of some inference rule in $\mathcal{I}$ to some subset of $\{\gamma_1, \ldots, \gamma_{i-1}\}$. If γ is an order dependency over U, then a *proof* of γ from Γ using $\mathcal{I}$ (relative to U) is a proof $\gamma_1, \ldots, \gamma_n$ from Γ using $\mathcal{I}$, where $\gamma_n = \gamma$. An order dependency γ is *provable* from Γ using $\mathcal{I}$ (relative to U) if there is a proof of γ from Γ using $\mathcal{I}$. In this case we write $\Gamma \vdash_{\mathcal{I},U} \gamma$ (with $\mathcal{I}$ and/or U omitted if understood from the context).

The 'correctness' of a set of inference rules is formally described by the following definition.

Definition. A set $\mathcal{I}$ of inference rules is *sound* if for each od-schema (U, Γ) and order dependency γ over U, $\Gamma \vdash \gamma$ implies $\Gamma \models \gamma$. $\mathcal{I}$ is *complete* if, for each od-schema (U, Γ) and order dependency γ over U, $\Gamma \models \gamma$ implies $\Gamma \vdash \gamma$.

To state the inference rules, the following will be used.

Definition. A marked attribute $\hat{A}$ is *linear* if it has the form A, $\tilde{A}$, $\bar{A}$, $\underline{A}$ or $\underline{\bar{A}}$ for some attribute A. A set M of marked attributes is *linear* if each element of M is linear.

Thus, for sets M and N of marked attributes, $M \rightarrow N$ is an order dependency iff MN is linear.

We now list seven (OD1–OD7) inference rules for order dependency. (Additional inference rules are presented at the end of the section.) It will be shown below that these seven rules form a sound and complete set.

In what follows, M, N, P, Q range over sets of linear marked attributes, and A ranges over individual attributes.

OD1 (*Reflexivity*) If $N \subseteq M$, then $M \rightarrow N$.

OD2 (*Augmentation*) If $M \rightarrow N$ and $Q \subseteq P$, then $MP \rightarrow NQ$.

OD3 (*Transitivity*) If $M \rightarrow N$ and $N \rightarrow P$, then $M \rightarrow P$.

OD4 (*Reversal*) If $M \rightarrow N$, then $M^R \rightarrow N^R$.

OD5 (*Disjunction*) If $M\tilde{A} \rightarrow N$ and $MA \rightarrow N$, then $M\bar{A} \rightarrow N$.

OD6 (*Total order*) If A has total order, and if $M\tilde{A} \rightarrow N$, $MA \rightarrow N$ and $M\underline{A} \rightarrow N$ then $M \rightarrow N$.

OD7 (*Impropriety*) For each linear marked attribute $\hat{B}$, if $M = \text{supp}(M)$ and $M \rightarrow \tilde{A}$, then $\emptyset \rightarrow \hat{B}$.

Rules (OD1)–(OD3) are analogous to ones for functional dependencies [4]. The fifth deals with the 'meaning' of $\bar{A}$ and the sixth incorporates a particular feature of attributes with total order. The seventh rule treats those cases where the original od-schema (U, Γ) is improper. (The intuition behind this last rule will be discussed after Proposition 4.1 below.)

We now demonstrate the soundness of the above set of inference rules.

Notation. Let $\mathcal{I} = \{OD1, \ldots, OD7\}$.

Proposition 4.1. *The set $\mathcal{I}$ is sound.*

Proof. Let (U, Γ) be an od-schema and $\gamma_1, \ldots, \gamma_n$ a proof from Γ using $\mathcal{I}$. To show that $\Gamma \models \gamma_n$ it suffices to show for each i, $1 \leq i \leq n$, that $\Gamma \cup \{\gamma_1, \ldots, \gamma_{i-1}\} \models \gamma_i$. Let i be in $\{1, \ldots, n\}$. If γ_i is in Γ or follows from any one of OD1, ..., OD6, then it is easily seen that $\Gamma \cup \{\gamma_1, \ldots, \gamma_{i-1}\} \models \gamma_i$.

Now suppose that γ_i follows from OD7. Then $\gamma_i = M \rightarrow \tilde{A}$ and $\gamma_i = \emptyset \rightarrow \hat{B}$ for some $j < i$, where $M = \text{supp}(M)$, A is in U and $\hat{B}$ is a marked attribute. Suppose that (U, I) satisfies $\Gamma \cup \{\gamma_1, \ldots, \gamma_{i-1}\}$. Then $I \models M \rightarrow \tilde{A}$, and hence I is empty. therefore $I \models \emptyset \rightarrow \hat{B}$, and $\Gamma \cup \{\gamma_1, \ldots, \gamma_{i-1}\} \models \gamma_i$ as desired. □

The intuition behind OD7 can now be explained in the context of the above proof. Suppose that for some M, where $M = \text{supp}(M)$, $M \rightarrow \tilde{A}$ arises in a proof from Γ. Consequently, only the empty instance satisfies Γ, i.e., Γ is improper. From this it immediately follows that $\Gamma \models \emptyset \rightarrow \hat{B}$ for each linear marked attribute $\hat{B}$.

We now turn to proving the completeness of $\mathcal{I}$. In fact, we demonstrate two results, namely, that (a) $\mathcal{I}$ is complete, and (b) $\mathcal{I} - \{OD7\}$ is 'complete' if only proper od-schemas are considered. Formally, we have the following.

Notation. Let $\mathcal{I}' = \{OD1, \ldots, OD6\}$.

Theorem 4.2. (a) *The set $\mathcal{I}$ is complete.*

(b) *For each proper od-schema (U, Γ) and order dependency γ over U, if $\Gamma \models \gamma$, then $\Gamma \vdash_{\mathcal{I}'} \gamma$.*

The argument requires a series of lemmas. The first concerns inferences from improper sets of order dependencies.

Lemma 4.3. *Let (U, Γ) be an improper od-schema and γ be an order dependency. Then, $\Gamma \vdash_{\mathcal{I}} \gamma$.*

Proof. Let $\gamma = P \rightarrow Q$, where $Q = \hat{C}_1 \cdots \hat{C}_n$ (each $\hat{C}_i$ a linear marked attribute), and let u be in Tup(U). Since Γ is improper, $\{u\} \not\models \Gamma$ by Proposition 1.3. Thus, there is some $M \rightarrow N$ in Γ such that $u[M]u$ is true but $u[N]u$ is false. Since $u[M]u$, $M \subseteq U = \text{comp}(U, U)$ by Lemma 2.5. By OD1, $\Gamma \vdash_{\mathcal{I}} U \rightarrow M$. Since $u[N]u$ is false, either (α) N is inconsistent, or (β) N is consistent and $\tilde{A}$ or $\underset{\sim}{A}$ is in N for some attribute A. (If N only contains marked attributes of the form A, $\bar{A}$ and $\underset{\sim}{A}$, then $u[N]u$ obviously holds.)

We now show that

$$\Gamma \vdash_{\mathscr{I}} \emptyset \to \hat{C}_i \quad \text{for each } i,\ 1 \leq i \leq n. \tag{4.1}$$

Suppose (α) holds. Select some A in U. Then $\tilde{A} \subseteq N$, since there are no tuples u_1 and u_2 such that $u_1[N]u_2$. By OD1, $\Gamma \vdash_{\mathscr{I}} N \to \tilde{A}$. Since $\Gamma \vdash_{\mathscr{I}} U \to M$, $\Gamma \vdash_{\mathscr{I}} M \to N$ and $\Gamma \vdash_{\mathscr{I}} N \to \tilde{A}$; $\Gamma \vdash_{\mathscr{I}} U \to \tilde{A}$ by two applications of OD3. Clearly, $U = \text{supp}(U)$. Thus, by OD7, $\Gamma \vdash_{\mathscr{I}} \emptyset \to \hat{C}_i$ for each i. Now suppose (β) holds. Assume $\tilde{A}$ is in N for some attribute A. Then $\tilde{A} \subseteq N$, and again $\Gamma \vdash_{\mathscr{I}} N \to \tilde{A}$ by OD1. As before, it follows that $\Gamma \vdash_{\mathscr{I}} \emptyset \to \hat{C}_i$ for each i. Finally, assume $\underset{\sim}{A} \subseteq N$. Then a similar argument (using OD4 as well as OD1 and OD3) shows that $\Gamma \vdash_{\mathscr{I}} \emptyset \to \hat{C}_i$ for each i. Thus (4.1) holds.

Since $\emptyset \subseteq P$, it follows from (4.1), OD1 and OD3 that $\Gamma \vdash_{\mathscr{I}} P \to \hat{C}_i$ for each i. Then $\Gamma \vdash_{\mathscr{I}} P \to \hat{C}_1 \cdots \hat{C}_n$. (By induction, suppose $\Gamma \vdash_{\mathscr{I}} P \to \hat{C}_1 \cdots \hat{C}_j$. Since $\hat{C}_{j+1} \subseteq \hat{C}_{j+1}$, $\Gamma \vdash_{\mathscr{I}} P\hat{C}_{j+1} \to \hat{C}_1 \cdots \hat{C}_{j+1}$ by OD2. Since $\Gamma \vdash_{\mathscr{I}} P \to \hat{C}_{j+1}$ and $P \subseteq P$, $\Gamma \vdash_{\mathscr{I}} P \to P\hat{C}_{j+1}$ by OD2. Then $\Gamma \vdash_{\mathscr{I}} P \to \hat{C}_1 \cdots \hat{C}_{j+1}$ by OD3.) □

In view of the above lemma, to complete the proof of Theorem 4.2 it suffices to show (b). The next lemma indicates that we need only consider order dependencies with one marked attribute on the right.

Lemma 4.4. *Let (U, Γ) be an od-schema and $\Gamma' = \{M \to \hat{A} \mid M \to N \text{ in } \Gamma, \hat{A} \text{ in } N\}$. Suppose that for each order dependency $P \to \hat{C}$ (where $\hat{C}$ is a linear marked attribute), $\Gamma' \models P \to \hat{C}$ implies $\Gamma' \vdash_{\mathscr{I}'} P \to \hat{C}$. Then for each order dependency $P \to Q$, $\Gamma \models P \to Q$ implies $\Gamma \vdash_{\mathscr{I}'} P \to Q$.*

Proof. First note that

$$\Gamma \vdash_{\mathscr{I}'} M \to \hat{A} \quad \text{for each order dependency } M \to \hat{A} \text{ in } \Gamma'. \tag{4.2}$$

(For let $M \to \hat{A}$ be in Γ'. Then, for some N, $M \to N$ is in Γ and $\hat{A}$ is in N. Since $\hat{A}$ is in N, $\hat{A} \subseteq N$ so $\Gamma \vdash_{\mathscr{I}'} N \to \hat{A}$ by OD1. Then $\Gamma \vdash_{\mathscr{I}'} M \to \hat{A}$ by OD3.) Also, it is clear that $\Gamma \models \Gamma'$ and $\Gamma' \models \Gamma$.

Now suppose $\Gamma \models P \to Q$. If $Q = \emptyset$, then $\Gamma \vdash_{\mathscr{I}'} P \to Q$ by OD1. Assume $Q = \hat{C}_1 \cdots \hat{C}_n$, $n \geq 1$, each $\hat{C}_i$ a linear marked attribute. Let i be in $\{1, \ldots, n\}$. Obviously, $\Gamma \models P \to \hat{C}_i$. Since $\Gamma' \models \Gamma$, $\Gamma' \models P \to \hat{C}_i$. By assumption, $\Gamma' \vdash_{\mathscr{I}'} P \to \hat{C}_i$. By (4.2), $\Gamma \vdash_{\mathscr{I}'} M \to \hat{A}$ for each order dependency $M \to \hat{A}$ in Γ'. It immediately follows that $\Gamma \vdash_{\mathscr{I}'} P \to \hat{C}_i$. Thus $\Gamma \vdash_{\mathscr{I}'} P \to \hat{C}_i$ for each i. Finally, $\Gamma \vdash_{\mathscr{I}'} P \to \hat{C}_1 \cdots \hat{C}_n$, as in Lemma 4.3. □

We now fix a particular od-schema and an order dependency logically implied by it.

Notation. Let (U, Γ) be a given proper od-schema, let $\Gamma' = \{M \to \hat{A} \mid M \to N \text{ in } \Gamma, \hat{A} \text{ in } N\}$, and let $P \to \hat{C}$ (where $\hat{C}$ is a linear marked attribute) be an order dependency such that $\Gamma \models P \to \hat{C}$.

Because of Lemma 4.4, the proof of Theorem 4.2 will be completed by showing that $\Gamma' \vdash_{\mathcal{I}} P \to \hat{C}$. To accomplish this, we present a lengthy development based on a particular context-free grammar G and five lemmas. (We assume that the reader is familiar with the concepts of context-free grammar and the languages they generate [15, 20].) In what follows, we introduce and use certain components of G. Later, we shall define G itself.

Suppose u and v in Tup(U) are such that (i) $\{u, v\} \models \Gamma$, and (ii) $u[P]v$. Since $\Gamma \models P \to \hat{C}$, $u[\hat{C}]v$. We now introduce objects called *descriptors* for analyzing pairs u, v which satisfy (i) and (ii). (The descriptors will be the variables of the context-free grammar G.)

Definition. A *descriptor* is a quadruple $\langle K, L, M, \Sigma \rangle \neq \langle \emptyset, \emptyset, \emptyset, \emptyset \rangle$, where

(a) K is a set of linear attribute comparators,

(b) L is a set of linear marked attributes,

(c) M is a set of attributes, each with total order, and

(d) Σ is a set of order dependencies, each with a single marked attribute on the right.

Given a descriptor $\langle K, L, M, \Sigma \rangle$, K may contain more than one attribute comparator over a given attribute and hence be inconsistent. Likewise, L may be inconsistent. Also, since U is finite, the set of descriptors (over U) is finite.

The following provides a mechanism for using descriptors to describe the relationship between a pair of tuples.

Notation. For each descriptor $S = \langle K, L, M, \Sigma \rangle$, let $\phi(S)$ be the order formula[22]

$$\phi(K) \wedge \phi(L) \wedge (\bigwedge \{\delta(A) \mid A \text{ in } M\}) \wedge (\bigwedge \{\phi(N \to \hat{B}), \phi(N^R \to \hat{B}^R) \mid N \to \hat{B} \text{ in } \Sigma\}).$$

Suppose that $S = \langle K, L, M, \Sigma \rangle$ is a descriptor and Σ is a proper set of order dependencies. Then for u, v in Tup(U), $u[\phi(S)]v$ iff $u[K]v$, $u[L]v$ and $\{u, v\} \models \Sigma$. (The significance of M will become apparent shortly.)

The start variable of our grammar G is now defined.

Notation. Let $S_0 = \langle K_0, L_0, M_0, \Sigma_0 \rangle$, where $K_0 = \emptyset$, $L_0 = P$, $M_0 = \{A \mid A \text{ has total order}\}$ and $\Sigma_0 = \Gamma'$.

To see that S_0 is a descriptor observe that either $P \neq \emptyset$ or $\Gamma' \neq \emptyset$, since $\Gamma' \models P \to \hat{C}$ by assumption.

The following is immediate (and the proof is omitted).

Lemma 4.5. *For each u and v in* Tup(U), $u[\phi(S_0)]v$ *iff* $\{u, v\} \models \Gamma$ *and* $u[P]v$.

[22] If either K, L, M, or Σ is empty, then the corresponding component(s) of this expression is (are) omitted. By definition of descriptor, at least one of them is nonempty.

In particular, $u[\phi(S_0)]v$ implies $u[\hat{C}]v$.

Suppose that $u[\phi(S_0)]v$. Then S_0 specifies certain characteristics concerning the relationship between u and v. However, it does not necessarily specify the exact relationship between u and v, i.e., comp(u, v). We now introduce formalism which allows us to calculate (essentially) all possible values of comp(u, v) for pairs u, v satisfying $u[\phi(S_0)]v$. This formalism will then be used to show that $\Gamma' \vdash_{\mathscr{S}} P \to \hat{C}$.

The terminal symbols of the grammar G are given in the following definition.

Definition. A descriptor $\langle K, L, M, \Sigma\rangle$ is *terminal* if either (a) K is inconsistent, or (b) K is consistent, $L = \emptyset$, $M = \emptyset$ and, for each order dependency $N \to \hat{B}$ in Σ, $N \not\subseteq K$ and $N^R \not\subseteq K$.

For an intuitive understanding of this notion, suppose $S = \langle K, L, M, \Sigma\rangle$ is terminal, $\{A \text{ in } U \mid A \text{ has total order}\} \subseteq \text{supp}(K)$ and K is consistent. Let u, v in Tup(U) be such that (i) $u[K]v$, and (ii) for each A not in supp(K), $u[\not A]v$. It is easily verified that $u[\phi(S)]v$, and hence that $\{u, v\} \models \Sigma$. Thus, S completely specifies a value for comp(u, v), namely $K \cup \{\not A \mid A \text{ in } U - \text{supp}(K)\}$, such that $\{u, v\} \models \Sigma$.

The symbol S_0 is not terminal. (To see this, suppose the contrary. Since $K_0 = \emptyset$, it is consistent. Also, $L_0 = P = \emptyset$, $M_0 = \{A \mid A \text{ has total order}\} = \emptyset$, and for each dependency $N \to \hat{B}$ in $\Sigma_0 = \Gamma'$, $N \not\subseteq K_0$ and $N^R \not\subseteq K_0$. Since $M_0 = \emptyset$, each attribute in U has empty or general-partial order. Hence, there exist u and v in Tup(U) such that $u[\not A]v$ for each A in U. Then $u[\phi(S_0)]v$. Since Γ' is proper, $\{u, v\} \models \Gamma'$ by Lemma 4.5. Since $P = L_0 = \emptyset$, $u[P]v$. And since $\hat{C}$ is linear, $u[\hat{C}]v$ is false. Thus $\{u, v\} \not\models P \to \hat{C}$, whence $\Gamma' \not\models P \to \hat{C}$, a contradiction.)

We are now ready for the grammar G.

Notation. Let $G = (V, T, R, S_0)$ be the context-free grammar in which

(i) V (the *total vocabulary*, i.e., the variables and the terminals) is the set of all descriptors,

(ii) T (the set of *terminals*) is the set of all terminal descriptors,

(iii) S_0 (the *start* variable) is as defined earlier, and

(iv) R (the set of *productions*) contains the following (for each consistent set K of linear attribute comparators, each pair L, N of sets of linear marked attributes, and each set M of attributes with total order):

(R1) for each linear attribute comparator $\hat{A}$, $\langle K, L\hat{A}, M, \Sigma\rangle \to \langle K\hat{A}, L - \{\hat{A}\}, M, \Sigma\rangle$;

(R2) for each attribute A,

(a) $\langle K, L\bar{A}, M, \Sigma\rangle \to \langle K\hat{A}, L - \{\bar{A}\}, M, \Sigma\rangle\langle KA, L - \{\bar{A}\}, M, \Sigma\rangle$,

(b) $\langle K, L\underline{A}, M, \Sigma\rangle \to \langle K\underline{A}, L - \{\underline{A}\}, M, \Sigma\rangle\langle KA, L - \{\underline{A}\}, M, \Sigma\rangle$;

(R3) for each attribute A with total order,

$\langle K, L, MA, \Sigma\rangle \to \langle K\hat{A}, L, M - \{A\}, \Sigma\rangle\langle KA, L, M - \{A\}, \Sigma\rangle\langle K\underline{A}, L, M - \{A\}, \Sigma\rangle$;

(R4) for each linear marked attribute $\hat{A}$,

(a) $\langle K, L, M, \Sigma \cup \{N \to \hat{A}\}\rangle \to \langle K, LA, M, \Sigma - \{N \to \hat{A}\}\rangle$
if $N \subseteq K$ and $N^R \not\subseteq K$,

(b) $\langle K, L, M, \Sigma \cup \{N \to \hat{A}\}\rangle \to \langle K, L\hat{A}^R, M, \Sigma - \{N \to \hat{A}\}\rangle$
if $N^R \subseteq K$ and $N \not\subseteq K$,

(c) $\langle K, L, M, \Sigma \cup \{N \to \hat{A}\}\rangle \to \langle K, L\hat{A}\hat{A}^R, M, \Sigma - \{N \to \hat{A}\}\rangle$
if $N \subseteq K$ and $N^R \subseteq K$.

Before continuing, we briefly indicate the intuition motivating the productions of G. Suppose that $S = \langle K, L, M, \Sigma\rangle$ is a nonterminal descriptor and that $u[\phi[S]]v$ for u, v in Tup(U). Speaking intuitively, the first component, K, of S gives very specific information about comp(u, v) (namely, that $K \subseteq \text{comp}(u, v)$). The other components give less specific information about comp(u, v). (E.g., if $\bar{A}$ is in L, then A or $\tilde{A}$ is in comp(u, v), and if A is in M, then exactly one of $\tilde{A}$, A or $\underset{\sim}{A}$ is in comp(u, v).) The productions of G constitute a mechanism for translating 'vague' information in the second, third and fourth components in a descriptor into more specific information in the first and second components.

We begin our formal treatment of the grammar G by noting four essential properties of the productions in R.

Lemma 4.6. *Let $S \to S_1 \cdots S_n$ be a production in R, where $S = \langle K, L, M, \Sigma\rangle$, $\Sigma \subseteq \Gamma''$, and $S_i = \langle K_i, L_i, M_i, \Sigma_i\rangle$ for each i. Then for each pair u, v in* Tup(U):

(a) $u[\phi(S)]v$ *iff* $u[\bigvee_{i=1}^{n} \phi(S_i)]v$,

(b) $\Sigma_i \subseteq \Sigma$ *for each* i,

(c) $M \subseteq (\text{supp}(KL) \cup M_i)$ *for each* i, *and*

(d) *if* $\Gamma'' \vdash_J K_iL_i \to \hat{C}$ *for each* i, *then* $\Gamma'' \vdash_J KL \to \hat{C}$.

Proof. Consider (a). The proofs that productions of type R1, R2 and R3 have the specified property are straightforward, and omitted. Suppose now that the production is of type R4(a). Thus $n = 1$, $K_1 = K$, $M_1 = M$ and, for some order dependency $N \to \hat{A}$, $N \to \hat{A}$ is in Σ, $N \subseteq K$, $N^R \not\subseteq K$, $\Sigma_1 = \Sigma - \{N \to \hat{A}\}$ and $L_1 = L\hat{A}$. Suppose further that u, v in Tup(U) satisfy $u[\phi(S)]v$. Then

$$u[\phi(K_1)]v \quad \text{(since } K_1 = K\text{)},$$

$$u[\phi(L)]v,\ u[\bigwedge\{\delta(B) \mid B \text{ in } M_1\}]v \quad \text{(since } M_1 = M\text{)}$$

and

$$u[\bigwedge\{\phi(Q \to \hat{B}), \phi(Q^R \to \hat{B}^R) \mid Q \to \hat{B} \text{ in } \Sigma_1\}]v \quad \text{(since } \Sigma_1 \subseteq \Sigma\text{)}.$$

Furthermore, since $N \subseteq K$, $u[K]v$ and $u[\phi(N \to \hat{A})]v$, we have $u[\hat{A}]v$. Therefore $u[\phi(L\hat{A})]v$, so $u[\phi(S_1)]v$ as desired. Next suppose that $u[\phi(S_1)]v$. In this case it clearly suffices to show that $u[\phi(N \to \hat{A}) \wedge \phi(N^R \to \hat{A}^R)]v$, since $K = K_1$, $L \subseteq L_1$, $M = M_1$ and $\Sigma = \Sigma_1 \cup \{N \to \hat{A}\}$. Since $u[\phi(L_1)]v$ and $\hat{A}$ is in L_1, $u[\hat{A}]v$. Thus $u[\phi(N \to \hat{A})]v$ holds.

We now demonstrate that $u[N^R]v$ is false, which implies $u[\phi(N^R \to \hat{A}^R)]v$. Obviously comp$(u, v)$ is consistent. Since $u[K]v$, $K \sqsubseteq$ comp(u, v) by Lemma 2.5. Then $K \subseteq$ comp(u, v) by the comment after Lemma 2.5. Thus supp(comp$(u, v) - K$) = supp(comp(u, v)) − supp(K). By (a) of the corollary to Lemma 2.4 (since K is consistent), supp(N^R) = supp$(N) \subseteq$ supp(K). Hence supp$(N^R) \cap$ supp(comp$(u, v) - K$) = $\emptyset$. By (b) of the corollary to Lemma 2.4, $N^R \subseteq K$ iff $N^R \sqsubseteq$ comp(u, v). By assumption, $N^R \not\subseteq$ K, whence $N^R \not\sqsubseteq$ comp(u, v). By Lemma 2.5, $u[N^R]v$ is false. This completes the proof for type R4(a).

The proofs for types R4(b) and R4(c) are analogous, and omitted.

Parts (b) and (c) of the lemma are immediate.

Finally, consider (d). Suppose $\Gamma' \vdash_{\mathscr{G}} K_i L_i \to \hat{C}$ for each i. If the production $S \to S_1 \cdots S_n$ is of type R1, then $\Gamma' \vdash_{\mathscr{G}} KL \to \hat{C}$ since $KL = K_1 L_1$. Suppose the production is of type R2(a). Then $n = 2$ and there is some attribute A such that $L = L'\bar{A}$, $L_1 = L_2 = L'$, $K_1 = K\tilde{A}$ and $K_2 = KA$, where $L' = L - \{\bar{A}\}$. By assumption, $\Gamma' \vdash_{\mathscr{G}} K\tilde{A}L' \to \hat{C}$ and $\Gamma' \vdash_{\mathscr{G}} KAL' \to \hat{C}$. By OD5, $\Gamma' \vdash_{\mathscr{G}} K\bar{A}L' \to \hat{C}$, i.e., $\Gamma' \vdash_{\mathscr{G}} KL \to \hat{C}$ as desired. If the production is of type R2(b), then a similar argument can be made, using OD4 and OD5. And if the production is of type R3, then the result follows from OD6.

Suppose the production is of type R4(a). Then $n = 1$, $K_1 = K$ and for some $N \to \hat{A}$ in Σ, $N \sqsubseteq K$, $N^R \not\sqsubseteq K$, $\Sigma_1 = \Sigma - \{N \to \hat{A}\}$ and $L_1 = L\hat{A}$. Since $N \sqsubseteq K$, $\Gamma' \vdash_{\mathscr{G}} K \to N$ by OD1. Because $N \to \hat{A}$ is in Σ and $\Sigma \subseteq \Gamma'$ by hypothesis, $\Gamma' \vdash_{\mathscr{G}} N \to \hat{A}$. By OD3, $\Gamma' \vdash_{\mathscr{G}} K \to \hat{A}$. Since $KL \sqsubseteq KL$, OD2 implies that $\Gamma' \vdash_{\mathscr{G}} KL \to KL\hat{A}$, i.e., $\Gamma' \vdash_{\mathscr{G}} KL \to K_1 L_1$. Since $\Gamma' \vdash_{\mathscr{G}} K_1 L_1 \to \hat{C}$ by assumption, $\Gamma' \vdash_{\mathscr{G}} KL \to \hat{C}$ by OD3 as desired. The arguments for types R4(b) and R4(c) are similar, and omitted. □

The preceding lemma is now applied to establish certain properties of words in the language $L(G)$ generated by G.

Lemma 4.7. *Let $S_0 \Rightarrow \cdots \Rightarrow S_1 \cdots S_n$ be a derivation in G, with each $S_i = (K_i, L_i, M_i, \Sigma_i)$ a terminal. Then for each pair u, v in* Tup(U):

(a) $u[\phi(S_0)]v$ *iff* $u[\bigvee_{i=1}^{n} \phi(S_i)]v$,

(b) $\Sigma_i \subseteq \Gamma'$ *for each* i,

(c) $M_0 \sqsubseteq$ (supp$(K_i L_i) \cup M_i$) *for each* i, *and*

(d) *if* $\Gamma' \vdash_{\mathscr{G}} K_i \to \hat{C}$ *for each* i, *then* $\Gamma' \vdash_{\mathscr{G}} P \to \hat{C}$.

Proof. Parts (a), (b) and (c) follow from the corresponding parts of Lemma 4.6. Consider (d). For each i, $\Gamma' \vdash_{\mathscr{G}} K_i L_i \to \hat{C}$ by OD2 since $\Gamma' \vdash_{\mathscr{G}} K_i \to \hat{C}$ by hypothesis. Using Lemma 4.6(d) and induction, $\Gamma' \vdash_{\mathscr{G}} K_0 L_0 \to \hat{C}$, i.e., $\Gamma' \vdash_{\mathscr{G}} P \to \hat{C}$. □

We next demonstrate a key property of the terminals appearing in words of $L(G)$.

Lemma 4.8. *Let $S_0 \Rightarrow \cdots \Rightarrow S_1 \cdots S_n$ be a derivation in G, with each $S_i = (K_i, L_i, M_i, \Sigma_i)$ a terminal. Then $\hat{C} \sqsubseteq K_i$ for each i.*

Proof. Let i be an integer, $1 \leq i \leq n$. If K_i is inconsistent, then $\hat{C} \sqsubseteq K_i$. Thus suppose K_i is consistent. Since $M_i = \emptyset = L_i$, $M_0 \subseteq \sup(K_i)$ by Lemma 4.7(c). Let $K_i' = K_i \cup \{\hat{A} \mid A \text{ in } U - \text{supp}(K_i)\}$. Then K_i' is consistent, since K_i is consistent and $M_0 = \{A \mid A \text{ has total order}\}$. Clearly, $\text{supp}(K_i') = U$. Let u, v in $\text{Tup}(U)$ satisfy $u[K_i']v$. Hence $\text{comp}(u, v) = K_i'$ by Lemma 2.5. Then $u[\phi(K_i')]v$. Since S_i is a terminal, $N \not\sqsubseteq K_i$ and $N^R \not\sqsubseteq K_i$ for each order dependency $N \rightarrow \hat{A}$ in Σ_i. Since N contains only linear marked attributes and $K_i' - K_i$ contains no such marked attributes, it follows from (b) of the corollary to Lemma 2.4 that $N \not\sqsubseteq K_i'$. Similarly, $N^R \not\sqsubseteq K_i'$ for each order dependency $N \rightarrow \hat{A}$ in Σ_i. By Lemma 2.5, $u[N]v$ and $u[N^R]v$ are both false. Thus, $u[\phi(N \rightarrow \hat{A}) \wedge \phi(N^R \rightarrow \hat{A})]v$ for each $N \rightarrow \hat{A}$ in Σ_i. Therefore $u[\phi(S_i)]v$, so that $u[\bigvee_{i=1}^{n} \phi(S_i)]v$. By Lemma 4.7(a), $u[\phi(S_0)]v$. By Lemma 4.5, $\{u, v\} \models \Gamma$ and $u[P]v$. Since $\Gamma \models P \rightarrow \hat{C}$, $u[\hat{C}]v$. Since $K_i' = \text{comp}(u, v)$, $\hat{C} \sqsubseteq K_i'$ by Lemma 2.5. Now $\hat{C}$ is a linear marked attribute and $K_i' - K_i$ contains no linear marked attributes. Thus $\hat{C} \sqsubseteq K_i$. □

Our last lemma here asserts the existence of a nonempty word in $L(G)$.

Lemma 4.9. *$L(G)$ contains at least one nonempty word.*

Proof. For each variable $S = \langle K, L, M, \Sigma \rangle$ in V, let $\mu(S) = (\#(L), \#(M), \#(\Sigma))$. Let $<$ be the partial order on triples of nonnegative integers defined by last difference, i.e., $(a, b, c) < (a', b', c')$ iff either $c < c'$, or $c = c'$ and $b < b'$, or $c = c'$ and $b = b'$ and $a < a'$. The proof (left to the reader) immediately results from the following four easily verified facts:

(1) For each production $S \rightarrow S_1 \cdots S_n$ in R, $\mu(S_i) < \mu(S)$ for each i.

(2) Each variable is on the left of at least one production in R.

(3) The right-hand side of each production in R is a nonempty word.

(4) Each sequence $\{(a_i, b_i, c_i)\}_{i=1}$ of tuples of nonnegative integers, with $(a_{i+1}, b_{i+1}, c_{i+1}) < (a_i, b_i, c_i)$ for each i, is finite. □

We are now able to establish the theorem.

Proof of Theorem 4.2. It suffices to show that $\Gamma' \vdash_{\mathcal{I}} P \rightarrow \hat{C}$. By Lemma 4.9 there is a nonempty word $S_1 \cdots S_n$ in $L(G)$, with $S_i = \langle K_i, L_i, M_i, \Sigma_i \rangle$ for each i. By Lemma 4.8, $\hat{C} \sqsubseteq K_i$ for each i. Thus $\Gamma' \vdash_{\mathcal{I}} K_i \rightarrow \hat{C}$ for each i (by OD1). Then $\Gamma' \vdash_{\mathcal{I}} P \rightarrow \hat{C}$ by Lemma 4.7(d). □

We conclude the section by listing some additional inference rules for order dependency. It is easily verified that each of these is sound.

The first two rules, analogues of OD5 and OD7, resp., are:

OD5′ (*Disjunction*) If $M\underset{\sim}{A} \rightarrow N$ and $MA \rightarrow N$, then $M\underset{\sim}{A} \rightarrow N$.

OD7′ (*Impropriety*) For each linear marked attribute $\hat{B}$, if $M = \text{supp}(M)$ and $M \rightarrow \underset{\sim}{A}$, then $\emptyset \rightarrow \hat{B}$.

The next three rules are analogous to functional dependency rules (FD4, FD5 and FD6 of [4]):

OD8 (*Pseudo-transitivity*) If $M \to N$ and $NP \to Q$, then $MP \to Q$.

OD9 (*Union*) If $M \to P$ and $M \to Q$, then $M \to PQ$.

OD10 (*Decomposition*) If $M \to PQ$, then $M \to P$ and $M \to Q$.

Certain special cases of OD1 are of note, specifically:

OD11 (a) For each attribute A,

$$A \to \bar{A}, \quad \tilde{A} \to \bar{A}, \quad A \to \underset{\sim}{A} \quad \text{and} \quad \underset{\sim}{A} \to \underset{\sim}{A}.$$

(b) If M is inconsistent and $\hat{B}$ is a linear marked attribute, then $M \to \hat{B}$.

Finally, we have:

OD12 (*Inconsistency*) For each linear marked attribute $\hat{B}$,

(a) if A has total or general-partial order, $M \to \tilde{A}$ and $M \to \underset{\sim}{A}$, then $M \to \hat{B}$; and

(b) if A has empty order and $M \to \tilde{A}$, then $M \to \hat{B}$.

OD13 (*Equality*) For each attribute A, $\bar{A}\underset{\sim}{A} \to A$.

OD14 (*Interchange*) Let A have total order and B be arbitrary.

(a) If $M\bar{A} \to \bar{B}$, then $M\underset{\sim}{B} \to \underset{\sim}{A}$.

(b) If $M\tilde{A} \to \tilde{B}$, then $M\underset{\sim}{B} \to \underset{\sim}{A}$.

(Analogous interchange rules can be obtained by using other combinations of $\tilde{A}$, $\bar{A}$, $\underset{\sim}{A}$, $\underset{\sim}{A}$, $\tilde{B}$, $\bar{B}$, $\underset{\sim}{B}$, $\underset{\sim}{B}$.)

5. Complexity

This section considers the complexity of determining logical implication for order dependency. The problem is shown to be NP-hard in general. More specifically, three restricted contexts are exhibited for which the problem of deciding "$\Gamma \models ? \to \hat{C}$" is NP-complete. In conclusion, another restricted context is presented for which the problem is polynomially decidable.

We now state the main result of the section.

Theorem 5.1. *It is co-NP-complete to determine whether* $\Gamma \models 0 \to \hat{C}$ (*where* $\hat{C}$ *is a linear marked attribute*) *in the following cases*:

(i) *if all relevant attributes have total order*;

(ii) *if all relevant attributes have total order and only marked attributes of the form* A, $\bar{A}$ *and* $\underset{\sim}{A}$ *appear in each order dependency*; *and*

(iii) *if all relevant attributes have general-partial order.*

It is clear that (ii) subsumes (i). However, we consider the proof of (i) first because it easily illustrates the general method used to demonstrate our NP-hardness results. After this, we sketch the modifications needed to prove (ii) and (iii). We assume throughout that the reader is faimiliar with the notions of NP-hardness and NP-completeness in general, and with 3SAT and reducibility in particular (see [14]).

For this section we use the following notation.

Notation. For each set Γ of order dependencies, let $|\Gamma|$ denote the length required to list the elements of Γ in succession. (For our purposes, if $\Gamma = \{M_1 \to N_1, \ldots, M_k \to N_k\}$, with no dependency repeated, then $|\Gamma| = 2k + \sum_{i=1}^{k} (\#(M_i) + \#(N_i))$.)

We start by showing that the problem of deciding whether $\Gamma \not\models P \to \hat{C}$ in cases (i), (ii) and (iii) is in NP. Indeed, suppose that

$$U = \mathrm{supp}(P\hat{C}) \cup \Big(\bigcup_{M \to N \text{ in } \Gamma} \mathrm{supp}(MN) \Big).$$

By Theorem 2.6(b), $\Gamma \not\models P \to \hat{C}$ iff Γ is proper and for some U-comparator K in $\mathscr{C}(\Gamma)$, $P \sqsubseteq K$ and $\hat{C} \not\sqsubseteq K$. By Proposition 1.3 it can be decided in polynomial time whether Γ is proper. It is easily seen that the procedure given in Section 2 for testing whether a given U-comparator is consistent is linear. And by Theorem 2.6(a) there is a polynomial time procedure to check whether a given consistent U-comparator is in $\mathscr{C}(\Gamma)$. Thus, a U-comparator K in $\mathscr{C}(\Gamma)$ such that $P \sqsubseteq K$ and $\hat{C} \not\sqsubseteq K$ can be 'guessed', and verification of these properties takes polynomial time. This implies that "$\Gamma \not\models P \to \hat{C}$?" is NP as desired.

In what follows we shall complete the proof of Theorem 5.1 by demonstrating that 3SAT, a problem known to be NP-complete [14], can be reduced (in polynomial time) to each of the problems (i), (ii) and (iii). To discuss 3SAT, we introduce the following notation.

Notation. A (*propositional*) *variable* is an abstract symbol. A (*propositional*) *literal* is a variable μ or a negated variable $\bar{\mu}$. A 3-*clause* is a set $\{\rho_1, \rho_2, \rho_3\}$ of exactly three literals. A set Π of 3-clauses is *satisfiable* if there is a truth assignment α for the variables occurring in Π such that in each 3-clause π of Π at least one literal in π is true under α.

We now introduce notation which permits us to translate instances of the 3SAT problem (i.e., sets of 3-clauses) into instances of a decision problem of the form "$\Gamma \not\models P \to \hat{C}$?".

Notation. Let $\Pi = \{\pi_1, \ldots, \pi_n\}$ be a fixed set of 3-clauses, with $\pi_i = \{\rho_1^i, \rho_2^i, \rho_3^i\}$. Let $\{\mu_1, \ldots, \mu_m\}$ be the set of those variables μ such that μ or $\bar{\mu}$ occurs in Π.

To prove Theorem 5.1(i), we translate Π as follows.

Notation. Let $U_1 = \{A\} \cup \{B_i, C_i \mid 1 \leq i \leq n\} \cup \{E_j \mid 1 \leq j \leq m\}$, with each attribute having total order. For each j, $1 \leq j \leq m$, let $E(\mu_j) = \tilde{E}_j$, $E(\bar{\mu}_j) = \underline{E}_j$ and

$$\Gamma_1 = \{\tilde{A} \to \tilde{B}_1 \cdots \tilde{B}_n\} \cup \{\tilde{B}_i\tilde{C}_i \to E(\rho_1^i), \tilde{B}_i C_i \to E(\rho_2^i), \tilde{B}_i \underline{C}_i \to E(\rho_3^i) \mid 1 \leq i \leq n\}.$$

Clearly, Γ_1 is proper (Proposition 1.3). Also, $|\Gamma_1|$ is linear in $\#(\Pi)$.

We now establish two lemmas, the second of which states that Π is satisfiable iff $\Gamma_1 \not\models \emptyset \to A$.

Lemma 5.2. *Let K be in $\mathscr{C}(\Gamma_1)$. Then $K \models \tilde{B}_i \to (E(\rho_1^i) \vee E(\rho_2^i) \vee E(\rho_3^i))$ for each i, $1 \leq i \leq n$.*

Proof. Let i be fixed. Since Γ_1 is proper and K is in $\mathscr{C}(\Gamma_1)$,

(1) $\quad K \models ((\tilde{B}_i \wedge \tilde{C}_i) \to E(\rho_1^i)) \wedge ((\tilde{B}_i \wedge C_i) \to E(\rho_2^i)) \wedge ((\tilde{B}_i \wedge \underset{\sim}{C}_i) \to E(\rho_3^i))$

by Proposition 3.3. Thus

(2) $\quad K \models [(\tilde{B}_i \wedge \tilde{C}_i) \vee (\tilde{B}_i \wedge C_i) \vee (\tilde{B}_i \wedge \underset{\sim}{C}_i)] \to [E(\rho_1^i) \vee E(\rho_2^i) \vee E(\rho_3^i)]$.

In turn, (2) yields

(3) $\quad K \models [\tilde{B}_i \wedge (\tilde{C}_i \vee C_i \vee \underset{\sim}{C}_i)] \to [E(\rho_1^i) \vee E(\rho_2^i) \vee E(\rho_3^i)]$.

Since C_i has total order, $K \models \hat{C}_i \vee C_i \vee \underset{\sim}{C}_i$. Thus,

(4) $\quad K \models \tilde{B}_i \to [\tilde{B}_i \wedge (\tilde{C}_i \vee C_i \vee \underset{\sim}{C}_i)]$.

Combining (3) and (4) we get $K \models \tilde{B}_i \to [E(\rho_1^i) \vee E(\rho_2^i) \vee E(\rho_3^i)]$. □

Lemma 5.3. *The following are equivalent:*

(a) $\Gamma_1 \not\models \emptyset \to A$.

(b) *Some comparator in $\mathscr{C}(\Gamma_1)$ contains $\tilde{A}$.*

(c) *Π is satisfiable.*

Proof. (a) $\Rightarrow$ (b): Suppose $\Gamma_1 \not\models \emptyset \to A$. By Theorem 2.6, $A \not\subseteq K$ for some U_1-comparator K in $\mathscr{C}(\Gamma_1)$. Since A has total order, either $\tilde{A}$ or $\underset{\sim}{A}$ is in K. If $\tilde{A}$ is in K, we are done. Suppose $\underset{\sim}{A}$ is in K. Then $\tilde{A}$ is in K^R and, since $\mathscr{C}(\Gamma_1)$ is closed under reversal, K^R is in $\mathscr{C}(\Gamma_1)$.

(b) $\Rightarrow$ (c): By assumption there exists a K in $\mathscr{C}(\Gamma_1)$ such that $\tilde{A}$ is in K. Let α be the truth assignment defined by $\alpha(\mu_i) =$ true if $K \models \tilde{E}_i$ and $\alpha(\mu_i) =$ false if $K \models \underset{\sim}{E}_i$. Since each attribute E_i has total order, α is well defined on each variable in Π. It is easily verified that for each literal ρ, $\alpha(\rho)$ is true iff $K \models E(\rho)$.

Since $\tilde{A}$ is in K, $K \models \tilde{A}$. Since $\tilde{A} \to \tilde{B}_1 \cdots \tilde{B}_n$ is in Γ_1, $K \models \phi(\tilde{A} \to \tilde{B}_1 \cdots \tilde{B}_n)$ by Proposition 3.3. Hence $K \models \tilde{B}_1 \cdots \tilde{B}_n$. By Lemma 5.2, $K \models E(\rho_1^i) \vee E(\rho_2^i) \vee E(\rho_3^i)$ for each i, $1 \leq i \leq n$. Hence, for each i, either $\alpha(\rho_1^i) =$ true or $\alpha(\rho_2^i) =$ true or $\alpha(\rho_3^i) =$ true. Therefore Π is satisfied by α.

(c) $\Rightarrow$ (a): Let α be a truth assignment which satisfies π. Let

$$K = \{\tilde{A}\} \cup \{\tilde{B}_i \mid 1 \leq i \leq n\} \cup \{\tilde{E}_j \mid 1 \leq j \leq m, \alpha(\mu_j) = \text{true}\}$$
$$\cup \{\underset{\sim}{E}_j \mid 1 \leq j \leq m, \alpha(\mu_j) = \text{false}\} \cup \{\tilde{C}_i \mid 1 \leq i \leq n, \alpha(\rho_1^i) = \text{true}\}$$
$$\cup \{C_i \mid 1 \leq i \leq n, \alpha(\rho_1^i) = \text{false}, \alpha(\rho_2^i) = \text{true}\}$$
$$\cup \{\underset{\sim}{C}_i \mid 1 \leq i \leq n, \alpha(\rho_1^i) = \text{false}, \alpha(\rho_2^i) = \text{false}, \alpha(\rho_3^i) = \text{true}\}.$$

Since α satisfies Π, exactly one attribute comparator over C_i is in K for each i. It readily follows that K is a U_1-comparator, and that K is in $\mathscr{C}(\Gamma_1)$. □

The above development shows that an arbitrary instance Π of 3SAT can be translated, in polynomial time, into a question of the form "$\Gamma \not\models P \to \hat{C}$?" (having the properties specified by Theorem 5.1(i)), where Π is satisfiable iff $\Gamma \not\models P \to \hat{C}$ is true. Thus, deciding if $\Gamma \not\models P \to \hat{C}$ in this context is NP-complete, and deciding if $\Gamma \models P \to \hat{C}$ is co-NP-complete. This concludes the proof of part (i) of the theorem.

Turning to part (ii) of Theorem 5.1, let Π be a set of 3-clauses as defined above.

Notation. Let $U_2 = \{B_i, C_i, D_i \mid 1 \leq i \leq n\} \cup \{E_j \mid 1 \leq j \leq m\} \cup \{F\}$ with each attribute having total order. For each j, $1 \leq j \leq m$, let $E'(\mu_j) = \bar{E}_j$, $E'(\bar{\mu}_j) = \underline{E}_j$ and

$$\Gamma_2 = \{\bar{B}_i\bar{C}_i\bar{D}_i \to E'(\rho_1^j), \bar{B}_i\underline{C}_i\underline{D}_i \to E'(\rho_2^j), \bar{B}_i\bar{C}_i\underline{D}_i \to E'(\rho_3^j), \bar{B}_i\underline{C}_i\bar{D}_i \to E'(\rho_4^j) \mid$$
$$\mid 1 \leq i \leq n\} \cup \{\bar{B}_i \to \bar{B}_k \mid 1 \leq i \leq n,\ 1 \leq k \leq n\} \cup \{E_i \to F \mid 1 \leq i \leq n\}.$$

Clearly Γ_2 is proper. The next lemma (the proof is omitted) plays the role of Lemma 5.3 in proving Theorem 5.1(ii).

Lemma 5.4. *The following are equivalent:*

(a) $\Gamma_2 \not\models \emptyset \to F$.

(b) *Some comparator in* $\mathscr{C}(\Gamma_2)$ *contains* $\tilde{F}$.

(c) *Some comparator in* $\mathscr{C}(\Gamma_2)$ *contains* $\tilde{E}_j$ *or* $\underline{E}_j$ *for each* j, $1 \leq j \leq m$.

(d) Π *is satisfiable.*

Obviously, Theorem 5.1(ii) readily results from Lemma 5.4.

Turning to part (iii), again let Π be a set of 3-clauses as above.

Notation. Let $U_3 = \{A\} \cup \{B_i, C_i, D_i \mid 1 \leq i \leq n\} \cup \{E_i \mid 1 \leq j \leq m\}$ with each attribute having general-partial order. Let $E''(\mu_j) = \tilde{E}_j$, $E''(\bar{\mu}_j) = E_j$ and

$$\Gamma_3 = \{\tilde{A} \to \bar{G} \mid G \text{ in } U_3\} \cup \{\tilde{A} \to \tilde{B}_1 \cdots \tilde{B}_n\}$$
$$\cup \{\tilde{B}_i\tilde{C}_i\tilde{D}_i \to E''(\rho_1^j), \tilde{B}_iC_iD_i \to E''(\rho_2^j),$$
$$\tilde{B}_i\tilde{C}_iD_i \to E''(\rho_3^j), \tilde{B}_iC_i\tilde{D}_i \to E''(\rho_4^j) \mid 1 \leq i \leq n\}.$$

Again, Γ_3 is proper. The verification of part (iii) of Theorem 5.1 results from the next lemma (the proof is omitted).

Lemma 5.5. *The following are equivalent:*

(a) $\Gamma_3 \not\models \emptyset \to A$.

(b) *Some comparator in* $\mathscr{C}(\Gamma_3)$ *contains* $\tilde{A}$.

(c) Π *is satisfiable.*

If all attributes have empty order, then the question "$\Gamma \models \sigma$?", where $\Gamma \cup \{\sigma\}$ is a set of order dependencies, essentially involves only functional dependencies and thus is decidable in linear time [2]. We conclude this section by presenting a more extensive context in which deciding the question "$\Gamma \models P \to \hat{C}$?" requires only polynomial time.[23]

Theorem 5.6. *Suppose that each relevant attribute has either empty or general-partial order, and that*

> *only marked attributes of the form A, $\bar{A}$ and $\underline{A}$ occur in each order dependency.* (5.1)

Then the question "$\Gamma \models P \to Q$?" is decidable in time a polynomial function of $|\Gamma \cup \{P \to Q\}|$.

Proof. Let Γ and $P \to Q$ be given, and let

$$U = \text{supp}(PQ) \cup \left(\bigcup_{M \to N \text{ in } \Gamma} \text{supp}(MN) \right).$$

By hypothesis, no attribute in U has total order. Since $\Gamma \models P \to Q$ iff $\Gamma \models P \to \hat{C}$ for each marked attribute $\hat{C}$ in Q, without loss of generality we may assume that Q is the singleton $\hat{C}$. (If deciding $\Gamma \models P \to \hat{C}$ requires time $f(|\Gamma \cup \{P \to \hat{C}\}|)$ for each $\hat{C}$ in Q, where f is a (monotonically nondecreasing) polynomial, then deciding $\Gamma \models P \to Q$ requires only

$$\#(Q) \cdot f(|\Gamma \cup \{P \to \hat{C}\}|) \leq |\Gamma \cup \{P \to Q\}| \cdot f(|\Gamma \cup \{P \to Q\}|),$$

a polynomial in $|\Gamma \cup \{P \to Q\}|$ as desired.) Also, note that Γ is proper because of the form of the dependencies in Γ.

Beginning with P, we now inductively build a particular set P^* of marked attributes. Let $P_0 = P$. Given P_i, if there is some dependency $M \to N$ in Γ such that $M \sqsubseteq P_i$ and $N \not\sqsubseteq P_i$, then set $P_{i+1} = P_i N$. Similarly, if $M^R \sqsubseteq P_i$ and $N^R \not\sqsubseteq P_i$ for some $M \to N$ in Γ, then set $P_{i+1} = P_i N^R$. Continue this process until no further changes can be made to P_i. Let P^* denote the last element obtained in the resulting sequence. It is easily seen that P^* is computed in time a polynomial of $|\Gamma \cup \{P \to Q\}|$. ($P_i$ can be expanded at most $2\#(\Gamma) \leq 2|\Gamma \cup \{P \to Q\}|$ times, and each such expansion requires time a linear function of $|\Gamma \cup \{P \to Q\}|$.)

Now let $K_1 = L_1 L_2 L_3$, where

$$L_1 = \{A \mid A \sqsubseteq P^*\}, \quad L_2 = \{\bar{A} \mid \bar{A} \sqsubseteq P^* \text{ and } \underline{A} \not\sqsubseteq P^*\}$$

and

$$L_3 = \{\underline{A} \mid A \sqsubseteq P^* \text{ and } \bar{A} \not\sqsubseteq P^*\}.$$

[23] The algorithm used to demonstrate this result is essentially a generalization of Algorithm 1 of [2]. (However, as will be seen, the proof of the validity of the algorithm is considerably more involved.) In fact, using the techniques of Algorithm 2 of [2], with a nontrivial amount of additional complication, our result can be strengthened from polynomial to linear running time.

Note that if A has empty order, then $\bar{A} \subseteq P^*$ implies $\underline{A} \subseteq P^*$ and conversely. Thus,

$$\text{if } A \text{ is in } \text{supp}(L_2L_3), \text{ then } A \text{ has general-partial order.} \tag{5.2}$$

Since P^* contains only marked attributes of the form A, $\bar{A}$ and $\underline{A}$, it readily follows that K_1 is a comparator (although not necessarily a U-comparator) and $\text{supp}(K_1) = \text{supp}(P^*)$. Let $K_2 = \{\hat{A} \mid A \text{ in } U\text{-supp}(P^*)\}$. Then $K = K_1K_2$ is a U-comparator. Clearly $P^* \subseteq K_1 \subseteq K$ and K is consistent. Also, K is computable in time a polynomial of $|\Gamma \cup \{P \to Q\}|$. (Obviously, K_1 can be computed in time a linear function of $\#(P^*)$, and hence in time a polynomial function of $|\Gamma \cup \{P \to Q\}|$.)

We now claim that

$$\begin{aligned}&\text{for each set } M \text{ containing only marked attributes of the form}\\ &A, \bar{A} \text{ or } \underline{A}, \text{ if } M \subseteq K, \text{ then } M \subseteq P^*.\end{aligned} \tag{5.3}$$

To see this, first note that $X \subseteq A$ for each attribute A and each $X \subseteq \{A, \bar{A}, \underline{A}\}$. Now suppose that $M \subseteq K$. If $M = \emptyset$, we are done. Suppose $M \neq \emptyset$. Let A be in $\text{supp}(M)$ and $M_A = \{\hat{A} \text{ in } M \mid \hat{A} \text{ over } A\}$. Two cases arise.

(α) A has empty order. Since $M_A \subseteq M \subseteq K$, $M_A \subseteq \{A, \bar{A}, \underline{A}\}$ and A is in $\text{supp}(M)$, $\hat{A} \subseteq K$ for at least one element in $\{A, \bar{A}, \underline{A}\}$. Since K is a U-comparator and (5.2) holds, A is in K. By definition of K, $A \subseteq P^*$. Hence $M_A \subseteq P^*$

(β) A has general-partial order. Four possibilities occur.

(i) A is in M_A. Then A is in K (since A has general-partial order). By definition of K, $A \subseteq P^*$. Therefore $\{A, \bar{A}, \underline{A}\} \subseteq P^*$, so $M_A \subseteq P^*$.

(ii) $\bar{A}$ and $\underline{A}$ are in M_A. Since $M_A \subseteq K$, A must be in K. As in (i), $M_A \subseteq P^*$.

(iii) $M_A = \{\bar{A}\}$. Then A or $\tilde{A}$ is in K. Then $A \subseteq P^*$ or $\bar{A} \subseteq P^*$. Since $\bar{A} \subseteq A$, $M_A = \{\bar{A}\} \subseteq P^*$ in either case.

(iv) $M_A = \{\underline{A}\}$. By an argument similar to (iii), we get $M_A \subseteq P^*$.

Thus $M_A \subseteq P^*$ for each A in $\text{supp}(M)$. Since $M = \bigcup_{A \text{ in } \text{supp}(M)} M_A$, $M \subseteq P^*$ and (5.3) holds.

We next show that K is in $\mathscr{C}(\Gamma)$. Suppose that $M \to N$ is in Γ and $M \subseteq K$. By (5.3) and the form of the order dependencies in Γ, $M \subseteq P^*$. By definition of P^*, $N \subseteq P^* \subseteq K$. Similarly, if $M^R \subseteq K$, then $N^R \subseteq K$. Since Γ is proper and K is consistent, K is in $\mathscr{C}(\Gamma)$ by Theorem 2.6(a).

We now show that $\Gamma \models P \to \hat{C}$ iff $\hat{C} \subseteq K$. Suppose $\hat{C} \not\subseteq K$. Since $P \subseteq K$, $\Gamma \not\models P \to \hat{C}$ by Theorem 2.6(b). Now suppose that $\hat{C} \subseteq K$. Since $\hat{C}$ has the form A, $\bar{A}$ or $\underline{A}$ for some attribute A, $\hat{C} \subseteq P^*$ by (5.3). Let K' be a U-comparator in $\mathscr{C}(\Gamma)$ such that $P \subseteq K'$. Clearly K' is consistent. A simple induction based on Theorem 2.6(a) shows that $P^* \subseteq K'$. Then $\hat{C} \subseteq P^* \subseteq K'$. Since K' is an arbitrary element of $\mathscr{C}(\Gamma)$, $\Gamma \models P \to \hat{C}$ by Theorem 2.6(b). Thus $\Gamma \models P \to \hat{C}$ iff $\hat{C} \subseteq K$.

As noted earlier, K can be computed in time a polynomial function of $|\Gamma \cup \{P \to Q\}|$. Since checking whether $\hat{C} \subseteq K$ is linear in $|\Gamma \cup \{P \to Q\}|$, the result now follows. □

References

[1] W.W. Armstrong, Dependency structures of database relationships, *Proc. IFIP '74* (North-Holland, Amsterdam, 1974) 580–583.

[2] C. Beeri and P.A. Bernstein, Computational problems related to the design of normal form relational schemas, *ACM TODS* **4** (1) (1979) 30–59.

[3] C. Beeri, M. Dowd, R. Fagin and R. Statman, On the structure of Armstrong relations for functional dependencies, IBM Res. Rept. RJ 2901, San Jose, California, 1980.

[4] C. Beeri, R. Fagin and J.H. Howard, A complete axiomatization for functional and multivalued dependencies in database relations, *Proc. ACM SIGMOD Internat. Conf. on the Management of Data*, Toronto (1977) 47–61.

[5] P.A. Bernstein, Synthesizing third normal form relations, *ACM TODS* **1** (4) (1976) 277–298.

[6] J.L. Bentley, Multidimensional binary search trees used for associative searching, *Comm. ACM* **18** (9) (1975) 509–517.

[7] E.F. Codd, A Relational model of data for large shared data banks, *Comm. ACM* **13** (6) (1970) 377–387.

[8] E.F. Codd, Further normalization of the database relational model, in: *Courant Computer Science Symposia 6: Data Base Systems* (Prentice-Hall, Englewood Cliffs, NJ, 1971) 33–64.

[9] J. Dong and R. Hull, Applying approximate order dependency to reduce indexing space, in: *Proc. ACM SIGMOD Internat. Conf. Management of Data*, Orlando, Florida (1982) 119–127.

[10] K.P. Eswaran and D.D. Chamberlin, Functional specifications of a subsystem for data base integrity, *Proc. Internat. Conf. Very Large Databases*, Framingham, MA (1975) 48–68.

[11] R. Fagin, Multivalued dependencies and a new normal form for relational databases, *ACM TODS* **2** (3) (1977) 262–278.

[12] R. Fagin, Functional dependencies in a relational database and propositional logic, *IBM J. Res. Develop.* **21** (6) (1977) 534–544.

[13] R. Fagin, Horn clauses and database dependencies, *J. ACM* **29** (4) (1982) 952–985.

[14] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to NP-completeness* (Freeman, New York, 1979).

[15] S. Ginsburg, *The Mathematical Theory of Context-Free Languages* (McGraw-Hill, New York, 1966).

[16] S. Ginsburg and R. Hull, Characterizations for functional dependency and Boyce–Codd normal form families, *Theoret. Comput. Sci.* **27** (1, 2) (1983).

[17] S. Ginsburg and S. Zaiddan, Properties of functional dependency families, *J. ACM*, **29** (3) (1982) 678–698.

[18] J. Grant and B.E. Jacobs, On the family of generalized dependency constraints, *J. ACM*, **29** (4) (1982) 986–997.

[19] M.M. Hammer and D.J. McLeod, Semantic integrity in a relational data base system, *Proc. Internat. Conf. Very Large Databases*, Framingham, MA (1975) 25–47.

[20] M.A. Harrison, *Introduction to Formal Language Theory* (Addison-Wesley, Reading, MA 1978).

[21] A. Klug, On inequality tableaux, Computer Sciences Tech. Rept. #403, University of Wisconsin at Madison, 1980.

[22] J. Nievergelt, H. Hinterberger and K.C. Sevcik, The grid file: An adaptable, symmetric multi-key file structure, Tech. Rept. Institut für Informatic, ETH Zürich, 1981.

[23] Y. Sagiv, C. Delobel, D.S. Parker and R. Fagin, An equivalence between relational database dependencies and a subclass of propositional logic, *J. ACM* **28** (3) (1981) 435–453.

[24] D.W. Shipman, The functional data model and the data language DAPLEX, *ACM TODS* **6** (1) (1981) 140–173.

[25] A.M. Silva and M.A. Melkanoff, A method for helping discover the dependencies of a relation, in: H. Gallaire, J. Minker and J.-M. Nicolas, eds., *Advances in Database Theory, Vol. 1* (Plenum, New York, 1981).

[26] M. Stonebraker, Implementation of integrity constraints and views by query modification, in: *Proc. ACM SIGMOD Internat. Conf. Management of Data*, San Jose, CA (1975) 65–78.

[27] J.D. Ullman, *Principles of Database Systems* (Computer Science Press, Potomac, MD, 1980).

[28] M. Yannakakis and C.H. Papadimitriou, Algebraic dependencies, *Proc. 21st Ann. Symp. on Foundations of Computer Science* (1980) 328–332.

[29] C. Zaniolo, Analysis and design of relational schemata for database systems, Tech. Rept. UCLA-ENG-7769, Dept. of Comput. Sci., UCLA, 1976.

[30] A.V. Aho, Y. Sagio and J.D. Ullman, Equivalences among relational expressions, *SIAM J. Comput.* **8** (2) (1979) 218–246.