Errata Note

Erratum for Discovering Order Dependencies through Order Compatibility

Jaroslaw Szlichta
Ontario Tech University
Canada
jarek@ontariotechu.ca

Parke Godfrey
York University
Canada
godfrey@yorku.ca

Lukasz Golab
University of Waterloo
Canada
lgolab@uwaterloo.ca

Mehdi Kargar
Ryerson University
Canada
kargar@ryerson.ca

Divesh Srivastava
AT&T Labs-Research
United States
divesh@research.att.com

ABSTRACT

A number of extensions to the classical notion of functional dependencies have been proposed to express and enforce application semantics. One of these extensions is that of order dependencies (ODs), which express rules involving order. The article entitled "Discovering Order Dependencies through Order Compatibility" by Consonni et al., published in the EDBT conference proceedings in March 2019, investigates the OD discovery problem. The authors claim to prove that their OD discovery algorithm, OCDDISCOVER, is *complete*, as well as being significantly more efficient in practice than the state-of-the-art. They further claim that the implementation of the existing FASTOD algorithm (ours)—we shared our code base with the authors—which they benchmark against is flawed, as OCDDISCOVER and FASTOD report different sets of ODs over the same data sets.

In this rebuttal, we show that their claim of completeness is, in fact, *not* true. OCDDISCOVER's pruning rules are overly aggressive, and prune parts of the search space that contain legitimate ODs. This is the reason their approach appears to be "faster" in practice. Finally, we show that Consonni et al. misinterpret our set-based canonical form for ODs, leading to an incorrect claim that our FASTOD implementation has an error.

1 INTRODUCTION

Integrity constraints specify the intended semantics of dataset attributes. They are commonly used in a number of application areas, such as schema design, data integration, data cleaning, and query optimization [2]. Past work focused primarily on *functional dependencies* (FDs). In recent years, several extensions to the notion of an FD have been studied, including that of *order dependencies* (ODs) [3, 5–8, 10]. FDs cannot capture relationships among attributes with naturally *ordered* domains, such as over timestamps, numbers, and strings, which are common in business data [9]. For example, consider Table 1, which shows employee tax records in which `tax` is calculated as a `percentage` of `salary`. Both `tax` and `percentage` are non-decreasing with `salary`.

Order dependencies naturally express such semantics. For a second example from Table 1, the OD <`salary` *orders* `group`, `subgroup`> holds. When the table is sorted by `salary`, it is also then sorted by `group` (with ties broken by `subgroup`). However, <`salary` *orders* `subgroup`, `group`> does *not* hold. This illustrates that the *order* in which attributes appear in the OD matters.

Table 1: Table with employee information.

#	ID	yr	posit	bin	sal	perc	tax	grp	subg
t1	10	19	secr	1	5K	20%	1K	A	III
t2	11	19	mngr	2	8K	25%	2K	C	II
t3	12	19	direct	3	10K	30%	3K	D	I
t4	10	18	secr	1	4.5K	20%	0.9K	A	III
t5	11	18	mngr	2	6K	25%	1.5K	C	I
t6	12	18	direct	3	8K	25%	2K	C	II

The theory of order dependencies subsumes that of functional dependencies. Any FD can be *mapped* to an equivalent OD by prefixing the left-hand-side attributes onto the right-hand side [8, 10]. For example, if `salary` *functionally determines* `tax`, then `salary` *orders* `salary`, `tax`.

The purpose of this article is to refute the following claims in Consonni et al. [3].

(1) The authors present a definition of *minimality* for order compatibility dependencies (OCDs). An OCD is a more specific form of order dependency in which two lists of attributes order each other, when taken together [8]. They claim that their definition of minimality is *complete*; that is, from it, one can recover all valid OCDs that hold over a given table.
(2) Given their definition of minimal OCDs, Consonni et al. [3] propose an algorithm to *discover* ODs via OCDs, which has factorial complexity in the number of attributes. They claim to prove that their algorithm produces a canonically *complete* set of ODs. (That is, a *minimal* set of ODs with respect to their definition, from which all the ODs which hold over the data could purportedly be inferred.)
(3) The authors claim that their experimental evaluation illustrates an *error* in our *implementation* of OD discovery algorithm (FASTOD) [6, 7], which leads to discovering many additional—and, purportedly, incorrect—dependencies. In spite of this claim of an "implementation error" in the FASTOD implementation that we provided them, they support via benchmark experiments that their algorithm, OCDDISCOVER, outperforms our algorithm, FASTOD.

We show that each of these three claims is incorrect, in turn.

(1) The definition of minimality in Consonni et al. [3]—insofar as its intended purpose is a *canonical* form—is incorrect. Their "canonical" form does not allow for the inference of *all* OCDs. It misses an important subclass of OCDs (and, respectively, ODs), any dependency which has a common prefix on the *left* and *right* (that is, repeated attributes at the beginning of the dependency).

(2) The claim of completeness of the OD discovery algorithm in Consonni et al. [3] is incorrect, as it relies upon their incorrect notion of "minimal" OCDs. Their conjecture that their algorithm is complete is incorrect; it is incomplete.
(3) Consonni et al. [3] misinterpret our set-based canonical form for ODs [6, 7] (which is equivalent to the list-based canonical form for ODs). This leads the authors to confuse set-based OCDs with ODs. Their claim that our implementation has an error arises from this, and their belief that their approach is complete. Consonni et al. [3] conclude that their algorithm is faster in practice, despite being significantly worse in asymptotic complexity. This arises in their benchmark experiments, however, due to the fact that their algorithm is incomplete.

In Section 2, we provide basic definitions and canonical forms for ODs. In Section 3, we analyze the completeness of OD discovery. In Section 4, we discuss the experimental evaluation conducted by Consonni et al. [3]. We conclude in Section 5.

2 FOUNDATIONS

2.1 Background

We use the following notational conventions.

Table 2: Notational conventions.

- **Relations.** **R** denotes a *relation schema* and **r** denotes a specific *table* instance. Letters from the beginning of the alphabet, A, B and C, denote single *attributes*. Additionally, t and s denote *tuples*, and t_A denotes the value of an attribute A in a tuple t.
- **Sets.** Letters from the end of the alphabet, $\mathcal{X}$, $\mathcal{Y}$ and $\mathcal{Z}$, denote *sets* of attributes. Also, $t_{\mathcal{X}}$ denotes the *projection* of a tuple t on $\mathcal{X}$. $\mathcal{XY}$ is shorthand for $\mathcal{X} \cup \mathcal{Y}$. The empty set of attributes is denoted as $\{\}$.
- **Lists.** **X**, **Y** and **Z** denote *lists*. The empty list of attributes is represented as [ ]. List [A, B, C] denotes an explicit list. [A | **T**] denotes a list with the *head* A and the *tail* **T**. **XY** is shorthand for **X** concatenate **Y**. Set $\mathcal{X}$ denotes the *set* of elements in *list* **X**. $\mathbf{X}^p$ denotes any arbitrary permutation of list **X** or set $\mathcal{X}$. Given a set of attributes $\mathcal{X}$, for brevity, we state $\forall i$, X_i to indicate indices $[1, ..., i]$ that have valid ranges ($i \leq |\mathcal{X}|$).

We provide a summary of the relevant definitions. The operator '$\preceq_{\mathbf{X}}$' defines a *weak total order* over any set of tuples, where **X** denotes a list of attributes. Unless otherwise specified, numbers are ordered numerically, strings are ordered lexicographically and dates are ordered chronologically.

Definition 2.1. [6, 7] Let **X** be a list of attributes. For two tuples t and s, $\mathcal{X} \in \mathbf{R}$, $t \preceq_{\mathbf{X}} s$ if[1]
– $\mathbf{X} = [\,]$; or
– $\mathbf{X} = [A \,|\, \mathbf{T}]$ and $t_A < s_A$; or
– $\mathbf{X} = [A \,|\, \mathbf{T}]$, $t_A = s_A$, and $t \preceq_{\mathbf{T}} s$.
Let $t \prec_{\mathbf{X}} s$ if $t \preceq_{\mathbf{X}} s$ but $s \not\preceq_{\mathbf{X}} t$.

Next, we define order dependencies.

Definition 2.2. [3, 5–8, 10] Let **X** and **Y** be lists of attributes over a relation schema **R**. Table **r** over **R** *satisfies* an OD $\mathbf{X} \mapsto \mathbf{Y}$ ($\mathbf{r} \models \mathbf{X} \mapsto \mathbf{Y}$), read as **X** *orders* **Y**, if for all $t, s \in \mathbf{r}$, $t \preceq_{\mathbf{X}} s$ implies $t \preceq_{\mathbf{Y}} s$. $\mathbf{X} \mapsto \mathbf{Y}$ is said to *hold* for **R** ($\mathbf{R} \models \mathbf{X} \mapsto \mathbf{Y}$) if, for each admissible table instance **r** of **R**, table **r** satisfies $\mathbf{X} \mapsto \mathbf{Y}$. $\mathbf{X} \mapsto \mathbf{Y}$ is *trivial* if, for all **r**, $\mathbf{r} \models \mathbf{X} \mapsto \mathbf{Y}$. $\mathbf{X} \leftrightarrow \mathbf{Y}$, read as **X** and **Y** are *order equivalent*, if $\mathbf{X} \mapsto \mathbf{Y}$ and $\mathbf{Y} \mapsto \mathbf{X}$.

[1] By some conventions, "*iff*"—"if and only if—would be used here. The intent, in any case, is that the use of "if" defines completely the notion.

The OD $\mathbf{X} \mapsto \mathbf{Y}$ means that **Y** values are monotonically non-decreasing wrt **X** values. Thus, if a list of tuples is ordered by **X**, then it is also ordered by **Y**, but not necessarily vice versa.

Example 2.3. Consider Table 1 in which tax is calculated as a percentage of salary, and tax groups and subgroups are based on salary. Tax, percentage and group are not decreasing with salary. Furthermore, within the same group, subgroup is not decreasing with salary. Finally, within the same year, bin increases with salary. Thus, the following order dependencies hold in that table: [`salary`] ↦ [`tax`], [`salary`] ↦ [`percentage`], [`salary`] ↦ [`group`, `subgroup`] and [`year`, `salary`] ↦ [`year`, `bin`].

Definition 2.4. [3, 5, 8, 10] Two order specifications **X** and **Y** are *order compatible*, denoted as $\mathbf{X} \sim \mathbf{Y}$, if $\mathbf{XY} \leftrightarrow \mathbf{YX}$. ODs in the form of $\mathbf{X} \sim \mathbf{Y}$ are called order compatible dependencies (OCDs)

The empty list of attributes (i.e., [ ]) is order compatible with *any* list of attributes. There is a strong relationship between ODs and FDs. Any OD implies an FD, modulo lists and sets, however, not vice versa.

LEMMA 2.5. [8, 10] If $\mathbf{R} \models \mathbf{X} \mapsto \mathbf{Y}$ (OD), then $\mathbf{R} \models \mathcal{X} \to \mathcal{Y}$ (FD).

Also, there is a correspondence between FDs and ODs.

THEOREM 2.6. [8, 10] $\mathbf{R} \models \mathcal{X} \to \mathcal{Y}$ *iff* $\mathbf{X} \mapsto \mathbf{XY}$, for any list **X** over the attributes of $\mathcal{X}$ and any list **Y** over the attributes of $\mathcal{Y}$.

ODs can be violated in two ways.

THEOREM 2.7. [8, 10] $\mathbf{R} \models \mathbf{X} \mapsto \mathbf{Y}$ (OD) *iff* $\mathbf{R} \models \mathbf{X} \mapsto \mathbf{XY}$ (FD) and $\mathbf{X} \sim \mathbf{Y}$ (OCD).

We are now ready to explain the two sources of OD violations: *splits* and *swaps* [8, 10]. An OD $\mathbf{X} \mapsto \mathbf{Y}$ can be violated in two ways, as per Theorem 2.7.

Definition 2.8. [8, 10] A *split* wrt an OD $\mathbf{X} \mapsto \mathbf{XY}$ (FD) is a pair of tuples s and t such that $s_{\mathcal{X}} = t_{\mathcal{X}}$ but $s_{\mathcal{Y}} \neq t_{\mathcal{Y}}$.

Definition 2.9. [8, 10] A *swap* wrt $\mathbf{X} \sim \mathbf{Y}$ (OCD) is a pair of tuples s and t such that $s \prec_{\mathbf{X}} t$, but $t \prec_{\mathbf{Y}} s$.

Example 2.10. In Table 1, there are three splits with respect to the OD [`position`] ↦ [`position`, `salary`] because `position` does not functionally determine `salary`. The violating tuple pairs are $t1$ and $t4$, $t2$ and $t5$, and $t3$ and $t6$. There is a swap with respect to [`salary`] ∼ [`subgroup`], e.g., over the pair of tuples $t1$ and $t2$.

2.2 Canonical Forms

Consonni et al. [3] use a native list-based canonical form, which is based on decomposing an OD into a FD and an OCD [8, 10]. Recall that based on Theorem 2.7 "OD = FD + OCD", as $\mathbf{X} \mapsto \mathbf{Y}$ *iff* $\mathbf{X} \mapsto \mathbf{XY}$ (FD) and $\mathbf{X} \sim \mathbf{Y}$ (OCD). The authors exploit this relationship to guide their discovery algorithm through order compatibility. Since they use a list-based representation for ODs, this leads to factorial complexity of OD discovery in the number of attributes.

Expressing ODs in a natural way relies on lists of attributes, as in the SQL order-by statement. One might well wonder whether lists are inherently necessary. We provide a polynomial *mapping* of list-based ODs into *equivalent* set-based canonical ODs [6, 7]. The mapping allows us to develop an OD discovery algorithm that traverses a much smaller set-containment lattice (to identify candidates for ODs) rather than the list-containment lattice used in Consonni et al. [3].

Two tuples, t and s, are *equivalent* over a set of attributes $\mathcal{X}$ if $t_{\mathcal{X}} = s_{\mathcal{X}}$. An attribute set $\mathcal{X}$ partitions tuples into *equivalence classes* [4]. We denote the *equivalence class* of a tuple $t \in \mathbf{r}$ over a set $\mathcal{X}$ as $\mathcal{E}(t_{\mathcal{X}})$, i.e., $\mathcal{E}(t_{\mathcal{X}}) = \{s \in \mathbf{r} \mid s_{\mathcal{X}} = t_{\mathcal{X}}\}$. A *partition* of $\mathbf{r}$ over $\mathcal{X}$ is the set of equivalence classes, $\Pi_{\mathcal{X}} = \{\mathcal{E}(t_{\mathcal{X}}) \mid t \in \mathbf{r}\}$. For instance, in Table 1, $\mathcal{E}(t1_{\{\text{year}\}}) = \mathcal{E}(t2_{\{\text{year}\}}) = \mathcal{E}(t3_{\{\text{year}\}}) = \{t1, t2, t3\}$ and $\Pi_{\text{year}} = \{\{t1, t2, t3\}, \{t4, t5, t6\}\}$.

We now present a set-based *canonical form* for ODs.

Definition 2.11. [6, 7] An attribute A is a *constant* within each equivalence class over $\mathcal{X}$, denoted as $\mathcal{X}$: [ ] $\mapsto$ A, if $\mathbf{X}^p \mapsto \mathbf{X}^p\mathsf{A}$. Furthermore, two attributes, A and B, are order-compatible within each equivalence class wrt $\mathcal{X}$, denoted as $\mathcal{X}$: A $\sim$ B, if $\mathbf{X}^p\mathsf{A} \sim \mathbf{X}^p\mathsf{B}$. ODs of the form of $\mathcal{X}$: [ ] $\mapsto$ A and $\mathcal{X}$: A $\sim$ B are called *(set-based) canonical* ODs, and the set $\mathcal{X}$ is called a *context*.

Example 2.12. In Table 1, the attribute `bin` is a constant in the context of `position` (`posit`) written as {`position`}: [ ] $\mapsto$ `bin`, since $\mathcal{E}(t1_{\{\text{position}\}}) \models$ [ ] $\mapsto$ `bin`, $\mathcal{E}(t2_{\{\text{position}\}}) \models$ [ ] $\mapsto$ `bin` and $\mathcal{E}(t3_{\{\text{position}\}}) \models$ [ ] $\mapsto$ `bin`. Also, there is no swap between `bin` and `salary` in the context of `year`, i.e., {`year`}: `bin` $\sim$ `salary`, since $\mathcal{E}(t1_{\{\text{year}\}}) \models$ `bin` $\sim$ `salary` and $\mathcal{E}(t4_{\{\text{year}\}}) \models$ `bin` $\sim$ `salary`.

Based on Theorem 2.13 and Theorem 2.14, list-based ODs in the form of FDs and OCDs, respectively, can be mapped into equivalent set-based ODs.

THEOREM 2.13. [6, 7] $\mathbf{R} \models \mathbf{X} \mapsto \mathbf{XY}$ *iff* $\forall \mathsf{A} \in \mathbf{Y}$, $\mathbf{R} \models \mathcal{X}$: [ ] $\mapsto$ A.

THEOREM 2.14. [6, 7] $\mathbf{R} \models \mathbf{X} \sim \mathbf{Y}$ *iff* $\forall i, j$, $\mathbf{R} \models \{\mathsf{X}_1, .., \mathsf{X}_{i-1}, \mathsf{Y}_1, .., \mathsf{Y}_{j-1}\}$: $\mathsf{X}_i \sim \mathsf{Y}_j$.

A list-based OD can be mapped into an equivalent set of set-based ODs via a polynomial mapping.

THEOREM 2.15. [6, 7] $\mathbf{R} \models \mathbf{X} \mapsto \mathbf{Y}$ *iff* $\forall \mathsf{A} \in \mathbf{Y}$, $\mathbf{R} \models \mathcal{X}$: [ ] $\mapsto$ A and $\forall i, j$, $\mathbf{R} \models \{\mathsf{X}_1, .., \mathsf{X}_{i-1}, \mathsf{Y}_1, .., \mathsf{Y}_{j-1}\}$: $\mathsf{X}_i \sim \mathsf{Y}_j$.

Example 2.16. The OD [AB] $\mapsto$ [CD] can be mapped to the following equivalent canonical ODs: {A, B}: [ ] $\mapsto$ C, {A, B}: [ ] $\mapsto$ D, {}: A $\sim$ C, {A}: B $\sim$ C, {C}: A $\sim$ D, {A, C}: B $\sim$ D.

3 COMPLETENESS ANALYSIS

While the theoretical search space for FASTOD [6, 7] is $O(2^{|\mathbf{R}|})$, the search space for OCDDISCOVER [3] is $O(|\mathbf{R}|!)$, which is much larger as it traverses a lattice of attribute *permutations* (where $|\mathbf{R}|$ denotes the number of attributes over a relational schema $\mathbf{R}$). To mitigate the factorial complexity, the list-based algorithm in Consonni et al. [3] uses pruning rules. We show that, despite the authors' claim that their approach discovers a canonically complete set of ODs, their pruning rules lead to *incompleteness*.

Section 3 in Consonni et al. [3] addresses their completeness "proof" for their OD discovery algorithm. The authors introduce a notion of *minimality* of a set of dependencies which is incorrect. Herein, a set of dependencies is called *minimal*—as it is in previous work on FDs and ODs [4, 6, 7]—if *all* dependencies that logically hold over a relation schema $\mathbf{R}$ can be inferred from this minimal (canonical) set of dependencies.[2] That is, a set of dependencies $\mathcal{M}$ is *minimal* over a table $\mathbf{r}$, if $\{\mathbf{X} \mapsto \mathbf{Y} \mid \mathcal{M} \models \mathbf{X} \mapsto \mathbf{Y}\}$ is equivalent to $\{\mathbf{X} \mapsto \mathbf{Y} \mid \mathbf{r} \models \mathbf{X} \mapsto \mathbf{Y}\}$.

[2] In some previous work [1], minimal dependencies $\mathcal{M}$ also satisfy an additional condition that that no proper subset of $\mathcal{M}$ can be used to infer all dependencies.

Thus, one should be able to infer from a minimal set of dependencies via the inference rules (axioms), $\mathcal{I}$, *all* the dependencies that are valid over the given instance of the table. That is, $\{\mathbf{X} \mapsto \mathbf{Y} \mid \mathcal{M} \vdash_{\mathcal{I}} \mathbf{X} \mapsto \mathbf{Y}\}$ is equal to $\{\mathbf{X} \mapsto \mathbf{Y} \mid \mathbf{r} \models \mathbf{X} \mapsto \mathbf{Y}\}$. Consonni et al. [3] use the set of *sound* and *complete* OD inference rules, $\mathcal{I}$, from [9, 10].

Pruning applied by a dependency discovery algorithm, thus, must respect minimality. This allows for the *implicit* discovery of the full set of valid dependencies, and thus be deemed *complete*.

In [3], an attribute list is minimal if it has no embedded order dependency (the list of attributes is the shortest possible).

Definition 3.1. [3] An attribute list $\mathbf{X}$ is *minimal* if there is no other list of attributes $\mathbf{X}'$ such that:

- $\mathbf{X}'$ is smaller than $\mathbf{X}$, and
- $\mathbf{X} \leftrightarrow \mathbf{X}'$

Example 3.2. [A, B, A] is *not* minimal as [A, B, A] $\leftrightarrow$ [A, B].

It follows then that an OCD is minimal in [3] *if and only if* there are no repeated attributes in the OCD. That is, there are no repeated attributes within the *left* or within the *right* list of the minimal OCD, as each is a minimal attribute list, *and* there is no repeated attribute between *left* and *right*.

Definition 3.3. [3] An OCD $\mathbf{X} \sim \mathbf{Y}$ is *minimal* if

- $\mathbf{X}$ and $\mathbf{Y}$ are minimal attribute lists *and*
- $\mathcal{X} \cap \mathcal{Y} = \emptyset$.

Definition 3.3 of *minimality* with no permitted repeated attributes is at the heart of the incompleteness problem of [3], as it does not allow for the inference of all dependencies that are valid over the given table. Theorem 3.4 states this, that an OCD with a common prefix between *left* and *right* (repeated attributes) can hold over a table, while no OCD without repeated attributes holds. Our proof of Theorem 3.4 is by offering a counter-example to the completeness premise in [3].

THEOREM 3.4. $\mathbf{R} \not\models \mathbf{Y} \sim \mathbf{Z}$, $\mathbf{R} \not\models \mathbf{XY} \sim \mathbf{Z}$ and $\mathbf{R} \not\models \mathbf{Y} \sim \mathbf{XZ}$ do not imply $\mathbf{R} \not\models \mathbf{XY} \sim \mathbf{XZ}$

Proof

It suffices to construct a table in which the OCD of the form

- $\mathbf{XY} \sim \mathbf{XZ}$

holds, but OCDs

- $\mathbf{Y} \sim \mathbf{Z}$,
- $\mathbf{XY} \sim \mathbf{Z}$, and
- $\mathbf{Y} \sim \mathbf{XZ}$

do not.

Consider Table 3 constructed over attributes A, B and C. In Table 3, the OCD [A, B] $\sim$ [A, C] holds, but [B] $\sim$ [C], [AB] $\sim$ [C], and [B] $\sim$ [AC] do not. □

In [3], the authors only show—as is stated in Theorem 3.5 below—that OCDs of the form $\mathbf{XY} \sim \mathbf{XZ}$ can be derived from $\mathbf{Y} \sim \mathbf{Z}$ (Theorem 3.5 via Theorem 3.10 in [3]).

THEOREM 3.5. [3] If $\mathbf{R} \models \mathbf{Y} \sim \mathbf{Z}$, then $\mathbf{R} \models \mathbf{XY} \sim \mathbf{XZ}$

Theorem 3.5 in [3] is *true*. The flaw in the authors' logic is that this theorem proves only *one* direction (the "if" of an intended "if and only if"). The "only if" (not proved by the theorem) is implicitly assumed as *true*, while it assuredly is not. It follows that their claim of canonical completeness for their definition of minimal OCDs is incorrect (Section 3.3 in [3]). OCDs with common prefixes between their *left* and *right* attribute lists are *not* redundant, by Theorem 3.4. This leads to an *incomplete* approach for OD

Table 3: Incompleteness of OCDDISCOVER [3].

#	A	B	C
t1	0	0	1
t2	1	1	0
t3	2	3	2
t4	3	2	3

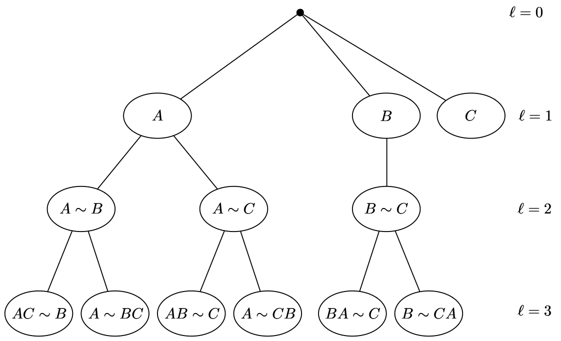

Figure 1: Lattice permutation tree for OCDDISCOVER [3].

discovery, as the recovery of the full set of valid dependencies is not possible.

Details of the OD discovery algorithm, OCDDISCOVER, by Consonni et al. [3] are presented in their Section 4. Let $\mathcal{U}$ be a set of attributes over a relation schema **R**. In the first level of the lattice, they generate candidates of the form $\mathsf{A} \sim \mathsf{B}$, where $\mathsf{A}, \mathsf{B} \in \mathcal{U}$ and $\mathsf{A} \neq \mathsf{B}$. (An OCD $\mathsf{B} \sim \mathsf{A}$ is not generated as it is equivalent to $\mathsf{A} \sim \mathsf{B}$.) At each level of the lattice (Fig. 1), if the candidate $\mathbf{X} \sim \mathbf{Y}$ is order compatible, they generate dependencies for the next level of the lattice. For each attribute not already present in the OCD, for each attribute $\mathsf{A} \in \mathcal{U} \setminus \{\mathcal{X} \cup \mathcal{Y}\}$, they add it to the right of each attribute list; i.e., $\mathbf{X}\mathsf{A} \sim \mathbf{Y}$ and $\mathbf{X} \sim \mathbf{Y}\mathsf{A}$. Thus, important OCDs with repeated attributes in a common prefix are never considered (as is consistent with their incorrect definition of minimality for OCDs). For example, an OCD [year, month] ∼ [year, week] would be missed. As a consequence, the authors do not discover ODs with repeated attributes, such as [year, salary] ↦ [year, bin] (recall Table 1).

In contrast, our FASTOD algorithm [6, 7] is *complete* for OD discovery. It does not miss dependencies with common prefixes. This is because the algorithm considers as candidates dependencies of the set-based form: OCD $\{\mathcal{X}\}$: $\mathsf{A} \sim \mathsf{B}$. This is built into the *context* of the set-based notation used in [6, 7], and cannot be missed when using this representation (see Theorem 2.14). Thus, dependencies with common prefixes are considered.

4 EXPERIMENTAL ANALYSIS

We demonstrate that the experimental analysis in Consonni et al. [3] that compares their OD discovery algorithm, OCDDISCOVER, with ours, FASTOD [6, 7], is incorrect. The authors misinterpret the set-based canonical representation for ODs as introduced in [6, 7] and as used in FASTOD. They conflate OCDs and ODs as we report them when evaluating the results. In [6, 7], we quantify the numbers of found FDs and OCDs. In [3], they incorrectly report these as the FDs and ODs, respectively, that we found. This occurs in their Table 6, where, for instance, they report 400 ODs and 89,571 FDs found by FASTOD, whereas this should be 400 OCDs and 89,571 FDs, respectively.

Table 4: Correctness of implementation for FASTOD [6].

#	A	B	C	D
t1	1	3	1	1
t2	2	3	3	2
t3	2	3	2	2
t4	2	5	2	2
t5	3	1	2	3
t6	4	4	4	2
t7	4	5	3	2

As a consequence of this misunderstanding of the set-based canonical representation for ODs [6, 7], the authors in [3] claim that the implementation of FASTOD finds ODs that are not present in the data. As an example of this, they provide the OD [B] ↦ [A, C] over Table 4 [3]. However, the FASTOD algorithm implementation in question finds the following ODs with respect to Table 4, where clearly the OD [B] ↦ [A, C] is not present.

(1) OCD {D}: A ∼ C
(2) OCD {C}: A ∼ D
(3) FD {A}:[ ] ↦ D
(4) OCD {B}: A ∼ D
(5) OCD {B}: C ∼ D
(6) OCD {B}: A ∼ C
(7) FD {B, C}:[ ] ↦ D
(8) FD {B, C}:[ ] ↦ A
(9) FD {A, B}:[ ] ↦ C
(10) OCD {C, D}: A ∼ B

The authors confuse the OCD {B}: A ∼ C with the OD [B] ↦ [A, C]. Consequently, they falsely assert that the reason the number of ODs found by OCDDISCOVER and FASTOD differ is due to an error in the implementation of FASTOD that we provided them.[3] The real reason that the number of reported dependencies differ, however, is, that OCDDISCOVER [3] is *incomplete*. The claim that they outperform the state-of-art despite a much worse asymptotic complexity, when tested in practice on real datasets, is invalid.

The authors in Consonni et al. [3] also state that FASTOD considers all columns to be of type string, while their code also considers real and integer numbers. While a minor point, we wish to clarify that the implementation we sent the authors does discover ODs over data types including real and integer numbers. The dependencies 1–10 reported in Table 4 remain the same, regardless of using numerical or string data type, given that the values are in the range of 1 to 5.

5 CONCLUSIONS

In this article, we have conducted a detailed analysis of the correctness of the results in the recent article by Consonni et al. [3] concerning the order dependency discovery problem. We have shown that, for the main claimed results related to the OD discovery problem, there are fundamental errors and omissions in the proof or experiments.

[3] While Consonni et al. [3] state that they were not able to isolate and resolve the root cause of what they felt was incorrect behavior in the implementation of FASTOD (which we had provided to them at their request for "ensuring fairness and reproducibility"), they never contacted us to help resolve it.

REFERENCES

[1] C. Beeri and P.A Bernstein. 1979. Computional Problems Related to the Design of Normal Form Relational Schemas. *TODS* 4, 1 (1979), 30–59.

[2] G. Cong, W. Fan, F. Geerts, X. Jia, and S. Ma. 2007. Improving Data Quality: Consistency and Accuracy. In *VLDB*. 315–326.

[3] C. Consonni, P. Sottovia, A. Montresor, and Y. Velegrakis. 2019. Discovering Order Dependencies through Order Compatibility. In *EDBT*. 409–420. https://doi.org/10.5441/002/edbt.2019.36

[4] Y. Huhtala, J. Karkkainen, P. Porkka, and H. Toivonen. 1998. Efficient Discovery of Functional and Approximate Dependencies Using Partitions. In *ICDE*. 392–401.

[5] P. Langer and F. Naumann. 2016. Efficient Order Dependency Detection. *VLDB J.* 25, 2 (2016), 223–241.

[6] J. Szlichta, P. Godfrey, L. Golab, M. Kargar, and D. Srivastava. 2017. Effective and Complete Discovery of Order Dependencies via Set-based Axiomatization. *PVLDB* 10, 7 (2017), 721–732.

[7] J. Szlichta, P. Godfrey, L. Golab, M. Kargar, and D. Srivastava. 2018. Effective and Complete Discovery of Bidirectional Order Dependencies via Set-based Axiomatization. *VLDB J.* 27, 4 (2018), 573–591.

[8] J. Szlichta, P. Godfrey, and J. Gryz. 2012. Fundamentals of Order Dependencies. *PVLDB, 5(11): 1220-1231* (2012).

[9] J. Szlichta, P. Godfrey, J. Gryz, W. Ma, W. Qiu, and C. Zuzarte. 2014. Business-Intelligence Queries with Order Dependencies in DB2. In *EDBT, 750-761*.

[10] J. Szlichta, P. Godfrey, J. Gryz, and C. Zuzarte. 2013. Expressiveness and Complexity of Order Dependencies. *PVLDB 6(14): 1858-1869* (2013).