How and Why False Denial Constraints are Discovered

Albert Martin
Universitat Politècnica de Catalunya
Barcelona, Spain
albert.martin.g@upc.edu

Eduardo C. de Almeida
Federal University of Paraná
Curitiba, Brazil
eduardo@inf.ufpr.br

Oscar Romero
Universitat Politècnica de Catalunya
Barcelona, Spain
oscar.romero@upc.edu

Anna Queralt
Universitat Politècnica de Catalunya
Barcelona, Spain
anna.queralt@upc.edu

ABSTRACT

Denial Constraints (DCs) are a flexible formalism to express many types of data rules, making them a widely adopted tool for many applications. This flexibility led to the development of numerous algorithms to automatically discover DCs directly from data. However, few studies have been conducted on the quality of the discovered DCs. We experimentally quantify the lack of quality in the results obtained by state-of-the-art algorithms, showing how the proportion of discovered DCs that are false is rarely below 95%. We hypothesize that the common source of these erroneous DCs stems from the adoption of the current DC validity definition. We use a statistical approach to explain the mechanism leading to these results, and propose a redefinition of DC validity properties to avoid the acceptance of false DCs. We validate this redefinition experimentally, showing that it exclusively accepts true constraints of the data, and is reliable enough to discover DCs missed by domain experts. Additionally, we provide curated sets of golden DCs for each dataset used in our study, those generated by domain experts and those discovered using our approach.

PVLDB Reference Format:
Albert Martin, Eduardo C. de Almeida, Oscar Romero, and Anna Queralt. How and Why False Denial Constraints are Discovered . PVLDB, 18(10): XXX-XXX, 2025.
doi:XX.XX/XXX.XX

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/nosocalgroc/DCValidity.

1 INTRODUCTION

Database data models provide constraints to define limitations on acceptable data values. These constraints can range from basic restrictions on attribute domain values to complex rules involving multiple predicates. Suppose we want to express the rule "No two records agree on the Social Security Number (SSN)" meaning the SSN must be unique within the database. This rule can be formalized as a Denial Constraint (DC) that is a mathematical notation that could be automatically manipulated to high-level languages. Informally, a DC is a statement of a situation that can not be true. For example, we express the SSN uniqueness constraint as the DC $\neg(t_x.SSN = t_y.SSN)$, where the predicate cannot be true for any pair of distinct records t_x and t_y. The main advantage of DCs lies in their flexibility, subsuming a wide variety of data constraints. This flexibility has made DCs a popular tool for various applications, including data cleaning [11, 27], integration [1], privacy [20], streaming [30], database design [19] and query optimization [25, 34]. As a result, datasets containing high-quality DCs are highly valuable for both research and practical applications.

Providing high-quality DCs is a challenging task, and their manual generation by domain experts proves to be expensive. This has prompted significant research into developing automated solutions [3, 5, 8, 17, 22–24, 26, 33], referred to as DC discovery algorithms. While these studies focus on improving the efficiency of DC discovery, they rarely analyze the quality of the discovered DCs. Some of the studies who do, comment on the discovery of overly contrived DCs that fail to represent any generalizable relationship of the data [8, 17, 23], and highlight how erroneous data can be one of the causes [8, 17, 23, 24, 33]. These are called overfit DCs, and Table 2 shows an example (φ_6) identified by all algorithms in a 15,000-record sample of the *Tax* dataset, with some records shown in Table 1. The main techniques proposed to avoid their discovery are ranking metrics that penalize superfluous complexity [8] and the acceptance of approximate DCs [17, 23, 24, 33]. However, these techniques promote the discovery of short and concise DCs that are not true [8, 23]. They are called underfit DCs, and Table 2 shows $\varphi_{(2-5)}$ as examples of underfit DCs accepted by all algorithms. Designing techniques to reduce the number of discovered DCs that are erroneous remains an open problem.

In this paper, we present a comprehensive experimental evaluation about the quality of the DCs discovered by several state-of-the-art algorithms. We provide results indicating that these algorithms discover thousands of long and contrived DCs, and how approximation is not able to reliably improve their quality. We also analyze how the discovered DCs compare to a set of golden DCs generated by domain experts, as done previously in the literature [8, 17]. This analysis indicates the proportion of discovered DCs that are false is often above 95%. We highlight how the techniques designed to prevent the acceptance of false DCs fail at improving the quality of the results.

Proceedings of the VLDB Endowment, Vol. 18, No. 10 ISSN 2150-8097.
doi:XX.XX/XXX.XX

Table 1: Tax data records.

TID	FN	LN	GD	AC	PH	CT	ST	ZIP	MS	CH	SAL	TR	STX	MTX	CTX
t1	Mark	Ballin	M	304	232-7667	Anthony	WV	25813	S	Y	5000	3	2000	0	2000
t2	Chunho	Black	M	719	154-4816	Denver	CO	80290	M	N	60000	4.63	0	0	0
t3	Annja	Rebizant	F	636	604-2692	Cyrene	MO	64739	M	N	40000	6	0	4200	0
t4	Annie	Puerta	F	501	378-7304	West Crossett	AR	72045	M	N	85000	7.22	0	40	0
t5	Anthony	Landram	M	319	150-3642	Gifford	IA	52404	S	Y	15000	2.48	40	0	40
t6	Mark	Murro	M	970	190-3324	Denver	CO	80251	S	Y	60000	4.63	0	0	0
t7	Ruby	Billinghurst	F	501	154-4816	Kremlin	AR	72045	M	Y	70000	7	0	35	1000
t8	Marcelino	Nuth	F	304	540-4707	Kyle	WV	25813	M	N	10000	4	0	0	0

First Name (**FN**), Last Name (**LN**), Gender (**GD**), Area Code (**AC**), Phone Number (**PH**), City (**CT**), State (**ST**), ZIP Code (**ZIP**), Marital Status (**MS**), Children Status (**CH**), Salary (**SAL**), Tax Rate (**TR**), Single Tax Exemption (**STX**), Marital Tax Exemption (**MTX**), Children Tax Exemption (**CTX**)

Table 2: Examples of DCs discovered from the 15000 first rows of the Tax dataset with the given approximation factors, and a comment on their meaning and trustworthiness.

DC	Aprox.	Semantics
$\varphi_1 : \neg(t_x.ST = t_y.ST \wedge t_x.SAL < t_y.SAL \wedge t_x.TR > t_y.TR)$	0%	True constraint of the data. One person cannot have lower salary (SAL) and higher tax rate (TR) than another on the same state (ST).
$\varphi_2 : \neg(t_x.ZIP = t_y.ZIP)$ $\varphi_3 : \neg(t_x.LN = t_y.LN)$ $\varphi_4 \neg(t_x.AC = t_y.AC)$ $\varphi_5 : \neg(t_x.ST = t_y.ST \wedge t_x.GD \neq t_y.GD)$	1%	Underfit constraints. It is easy to verify that ZIP codes (ZIP), last names (LN), and area codes (AC) are not unique, and the state where an individual lives (ST) clearly does not determine their gender (GD).
$\varphi_6 : \neg(t_x.GD = t_y.GD \wedge t_x.PH = t_y.PH \wedge t_x.CT = t_y.CT \wedge t_x.MS = t_y.MS \wedge t_x.CH \neq t_y.CH \wedge t_x.TR \neq t_y.TR \wedge t_x.STX = t_y.STX \wedge t_x.MTX \geq t_y.MTX \wedge t_x.CTX \leq t_y.CTX)$	0%	Overfit constraint. Too complex to represent any actual relationship of the data, in addition to being comprised of seemingly unrelated predicates.

We hypothesize that the common source of these erroneous DCs stems from the properties of the current DC validity definition [7, 8] adopted by the existing discovery algorithms. Our analysis shows that these properties are inadequate for rejecting sets of independent predicates (i.e. DCs that do not represent real relationships of the data), and we experimentally show how they often comprise over 30% of the output of the DC discovery algorithms. In light of this, we present a statistical approach to redefine the properties that constitute a valid DC. Our goal is to ensure no DC can be discovered without capturing some relationship of the data. We experimentally validate our redefined DC validity by showing how our modification decreases the number of discovered DCs that are not golden by over 95%. Furthermore, DCs valid under our new definition appear to be true constraints of the data even if they are not golden. This means our method generalizes well, as it is able to automatically detect complex true DCs missed by domain experts. In summary, our main contributions in this paper are as follows:

Experimental Analysis: We present a comprehensive experimental analysis of five DC discovery algorithms that rely on the current DC validity definition presented in Section 2. While we focus on these five algorithms due to space limitations, our findings can be generalized to any DC discovery algorithm that adheres to this DC validity definition. We empirically analyze the quality of the DCs discovered by these algorithms in Section 3 and partially explain them by statistically modeling sets of independent predicates in Section 4. We describe the full mechanism behind their discovery in Section 5, and show experimental results to support it in Section 6.

Definition of DC validity: We provide modifications to the current DC validity definition to ensure that the discovered DCs always correspond to true relationships between the data attributes in Section 5. Finally, we analyze the quality of the DCs that are valid under our new definition in Section 6.

Reproducibility and DC datasets: We provide source code to reproduce all the experiments with the main state-of-the-art DC discovery algorithms. The reproducibility artifacts include datasets, algorithms[1], compiled executables and saved intermediate results from the experiments. We also provide two sets of golden DCs for each dataset used, those generated by domain experts and those obtained using our approach.

In Section 2, we review the current DC definition and related work, and we present our conclusions in Section 7.

2 BACKGROUND

This section presents the original definition of a DC and the DC discovery problem, as formalised respectively in [7, 8]. We also provide an overview of a non-exhaustive list of DC discovery algorithms. To the best of our knowledge, all existing DC discovery algorithms in the literature adhere to the same definition and properties discussed in this section, resulting in similar outputs.

[1]Source codes for the algorithms have been modified to accept a common input format.

2.1 Denial Constraints

The fundamental components of DCs are predicates, which can be represented as boolean functions defined over pairs of tuples $P : DxD \rightarrow \{0, 1\}$. We consider predicates taking the form $P(t_x, t_y) = t_x.A\phi t_y.B$, with ϕ being an operator among the set $\{=, \neq, \geq, \leq, >, <\}$ and A and B attributes of relation D. Formally, a DC φ is a first-order logic formula that asserts no pair of distinct tuples can simultaneously satisfy all predicates P_i. This can be expressed as:

$$\forall t_x, t_y \in D, \neg(P_1 \wedge P_2 \wedge \cdots \wedge P_m)$$

This is a very flexible language, able to express a variety of rules, like Unique Column Combination (UCC) [4, 12, 15], Functional Dependencies (FDs) [15, 18, 21] and Order Dependencies (ODs) [16, 29]. For example, a UCC on attribute A would take the form $\neg(t_x.A = t_y.A)$, indicating no pair of tuples share the same value. A constant restriction would take the form $\neg(t_x.A \neq t_y.A)$, implying all values of the column are the same. More complex relationships may be represented, like the FD $A \rightarrow B$, as $\neg(t_x.A = t_y.A \wedge t_x.B \neq t_y.B)$, which implies it is impossible to find two tuples with the same determinant A and different dependent B. The OD $\neg(t_x.C < t_y.C \wedge t_x.D \geq t_y.D)$ states the tuple with a smaller value in attribute C also has a smaller value in attribute D.

2.2 Denial Constraint Discovery Problem

DCs subsume many kinds of integrity rules, making them highly valuable for numerous data processing tasks. However, it is essential to define what qualifies a set of predicates as a valid DC, thus determining which DCs are to be discovered and which are not. The properties that define a valid DC utilized in the DC discovery literature, are:

- **Satisfiability**: The DC must be satisfied for 100% of tuple pairs. To increase robustness against inconsistent data [17, 23, 24, 33], this condition may be relaxed by introducing an approximation factor ϵ, allowing the DC to be valid if this tuple pair satisfaction is above $100\% - \epsilon$.
- **Symmetry**: The satisfaction of a DC for a pair of tuples t_x and t_y is invariant to swapping t_x with t_y.
- **Minimality**: Removing a predicate from the DC makes it an invalid DC.
- **Non-triviality**: The DC is not satisfied by every possible data. This property prevents the acceptance of DCs like $\neg(P \wedge \neg P)$, which are always satisfied.

In [8], the DC discovery problem was formalized as the task of reporting all DCs that satisfy these properties within a given dataset.

2.3 Discovery Algorithms

Several algorithms have been published with the goal of discovering DCs from a given dataset [3, 5, 8, 17, 22–24, 26, 33]. Each of them presented novel approaches to optimize the discovery process, but they all share the common goal of discovering every set of predicates fulfilling the DC validity definition described in Section 2.2.

Indeed, the structure of the algorithms has evolved greatly from the initial discovery algorithm FastDC [8]. For instance, the Evidence Inversion algorithm from Hydra [5] offers an efficient alternative to FastDC's DFS when it comes to exploring the space of candidate exact DCs. FastADC [33] modifies the Hydra algorithm to accommodate approximation. Alternative techniques like Position List Indexes from DCFinder [23], MMCS based DC enumeration from ADCMiner [17] or sophisticated inverted indices from ECP [24] can be combined due to each optimizing different stages of the discovery process. Similarly important and novel improvements are presented in all other publications to greatly speed up the DC discovery, yet the *goal* of this search (the DC validity definition) has remained unchanged through the years of research. While our experimental analysis focuses on the algorithms whose source code we have been able to obtain, our findings and conclusions are applicable to all DC discovery algorithms based on this DC validity definition.

3 CHALLENGES IN DC DISCOVERY

The field of DC discovery has seen success in optimizing the efficiency of their algorithms. However, there has been little research analyzing the quality of the results yielded by these algorithms. Several publications mention how some discovered DCs fulfill all properties necessary to be deemed valid, but fail to represent any kind of useful data relationship [8, 22–24, 33]. This section summarizes the challenges the community has faced with respect to the quality of the discovered DCs, in addition to providing experimental evidence highlighting the severity of these issues.

3.1 Overfit DCs

In the context of DCs, overfitting refers to a DC being too specific to represent a general rule of the data [8]. Some previous work discussed that many discovered DCs are too complex[2] because they focus on a specific subset of the data instead of representing relationships that are general on the overall dataset. Table 2 shows an example of an overfit constraint discovered by all the previous algorithms on the *Tax* dataset, φ_6. While no definitive set of properties exists to unequivocally determine if a constraint is overfit, the general intuition is that a constraint is considered overfit when it is *overly complex*. The reason behind their appearance on clean data is provided in Section 4, and a full categorization and detection method for the general case is provided in Section 5.

Most publications presenting DC discovery algorithms talk about overfit DCs in the context of erroneous data [8, 22, 24, 33]. An intuitive example similar to the one presented in [33] is provided next: Consider the *Tax* dataset from Table 1, and let $\neg(t_x.CT = t_y.CT \wedge t_x.ST \neq t_y.ST)$ be a DC that holds on the data. If a new tuple containing an erroneous *City* (CT) or *State* (ST) is introduced, this DC will no longer hold, as there will be tuple pairs with the same city and different state. If the *First Name* (FN) attribute of this new tuple was not present in the data, the DC $\neg(t_x.CT = t_y.CT \wedge t_x.ST \neq t_y.ST \wedge t_x.FN = t_y.FN)$ will be accepted instead, as none of the tuple pairs that failed to fulfill the true DC will share names. This behavior generalizes for any possible source of erroneous values in the data, leading to the true DC being rejected

[2] In the context of DCs, the terms *complexity* and *generality* refer to the number of predicates of a DC [8, 17, 23]. As an intuitive example, DC φ_6 would be considered *complex/not general* because it consists of many predicates, restricting highly specific data states. On the other hand DC φ_2 is *not complex/general* as it contains only a single predicate, imposing less specific states of the data.

in favor of an alternative with additional random predicates, like the equality predicate on the FN attribute.

In practice, however, this is not the only form overfit DCs take, as they also appear for data completely free of errors. In fact, all algorithms yield large amounts of extremely long and contrived DCs, like φ_6. This DC seems too complex to represent a real relationship of the data, yet it lies within the current DC validity definition. These two types of erroneous DCs might seem to belong to entirely different families, since one appears exclusively for erroneous data, but later sections of this paper discuss how they share the same generation mechanism. While they both constitute "overly complex DCs", we will focus on overfit DCs on clean data at first, and deal with the general case where data may be erroneous in Section 5.

The presence of large amounts of these overfit DCs in the results has been pointed out in the first publication exploring the discovery of DCs [8]. It discusses how there is a large amount of long DCs that fulfill the DC validity definition yet have little usefulness, leading to sets of discovered DCs lacking in value. To address this issue, the concept of DC ranking was proposed to organize the DCs based on an objective metric and facilitate top-k ranking. The literature identified the following metrics for this purpose:

- **Succinctness**: Measures the complexity of a DC φ in terms of its length $Len(\varphi)$, as the ratio between the length of the DC and the length of the smallest DC: $\frac{Min\{(Len(\varphi')|\forall\varphi'\}}{Len(\varphi)}$. Length can be defined as the number of predicates or the number of symbols needed to express a DC. This metric penalizes overly complex constraints, preventing overfit DCs from ranking highly.
- **Coverage**: The satisfaction of a DC φ measures the proportion of tuple pairs (t_x, t_y) that do not satisfy all predicates at the same time as $\frac{\sum_{\forall t_x,t_y} \mathbb{1}_{\varphi(t_x,t_y)}}{\sum_{\forall t_x,t_y} 1}$. Coverage instead weights the contribution of every tuple pair by the proportion of satisfied predicates $w(t_x, t_y) = \frac{|\{P|P(t_x,t_y),P\in\varphi\}|+1}{|\varphi|}$, as $\frac{\sum_{\forall t_x,t_y} \mathbb{1}_{\varphi(t_x,t_y)} w(t_x,t_y)}{\sum_{\forall t_x,t_y} 1}$. This metric aims to measure statistical significance by penalizing DCs composed of predicates that are satisfied in isolation.
- **Interestingness**: Both Succinctness and Coverage were deemed desirable properties of discovered DCs, so this metric was defined to be a linear combination of the two in order to allow the user to choose the ideal balance for their particular case.

These metrics succeed in ranking long overfit constraints lower in the top-k, but fail to rank only the true constraints higher in the top-k [8, 23], as we show shortly in Section 3.3. The most commonly proposed solution to overfit constraints is the use of aproximate constraints [17, 23, 24, 33]. Approximate constraints would prevent overfitting by allowing the acceptance of more general rules, rather than introducing complexity to accommodate outliers or erroneous data. Furthermore, approximation has been shown to be a reliable method to deal with overfitting for many other Integrity Rule languages [6, 10, 13, 15, 31], proving its potential to improve the quality and usefulness of discovered DCs.

3.2 Underfit DCs

Previous work [8, 23] describes multiple short and concrete constraints, yet considered false, when ranking discovered DCs according to the previous metrics. They attribute their appearance on "under-fitting the data", hence the name, and examples like $\varphi_{(2-5)}$ from Table 2 are easy to find when running any algorithm with approximation. These DCs are concise enough to be easily interpretable as uniqueness constraints and functional dependencies, yet none of them are actually true. As before, there is no concrete set of rules able to qualify a DC as underfit, but the general intuition is that a DC is underfit if it is *overly simplistic*. The reason behind their discovery is provided in Section 4, and a full categorization and detection method is provided in Section 5.

Table 3: Succinctness and Coverage of underfit constraints from Table 2, showing how the ranking metrics proposed in [8] fail at only ranking high true DCs.

φ	$Succinctness(\varphi)$	$Coverage(\varphi)$
φ_2	1	0.99996
φ_3	1	0.9999
φ_4	1	0.9912
φ_5	0.5	0.75

Table 3 shows how all metrics proposed score high for underfit constraints, explaining why the experiments of [8] and [23] presented false constraints in the set of top-k highest ranked. There has been no further research using these metrics, as they do not appear to rank only the true constraints high in the top-k. This leaves the issue of filtering out underfit constraints as an open problem. In essence, the only accepted solution to deal with overfit constraints is approximation, but this leads to the acceptance of underfit constraints for which no solution has been presented yet.

3.3 Analysis of current algorithms

The presence of erroneous DCs is well known in the community, but they are referenced less frequently in the most recent publications. Research focuses on improving the efficiency of the discovery algorithms, rather than on the actual quality or usefulness of the discovered DCs. This is because there is no formal definition for what constitutes an overfit or underfit DC. One can look at a long/short DC that makes no sense and call it overfit/underfit, but there is no way to accurately classify a set of DCs into these categories according to a deterministic set of rules.

We now present experimental evidence to demonstrate that the quality of DC discovery is just as critical as discovery efficiency, as a lack of either can undermine the power of the formalism. We have collected the DCs discovered by several algorithms [5, 8, 17, 23, 33] on a variety of datasets and approximation factors. For every such combination, Figure 1 shows the proportion of discovered DCs with a specific number of predicates. Our experiments with no approximation reveal the same flaws discussed in the literature: over 95% of discovered DCs have 5 or more predicates for half of the datasets, and this proportion of long DCs is never below 20% for the remaining datasets. While it is impossible to claim that there exists no true constraint of this length, it is unlikely they account for such large amount of discovered DCs. The literature describes these long DCs as overly complex and too specific to a very small subset of the data, and presents approximation as a way to prevent them. However, our experiments show that this is not the general case,

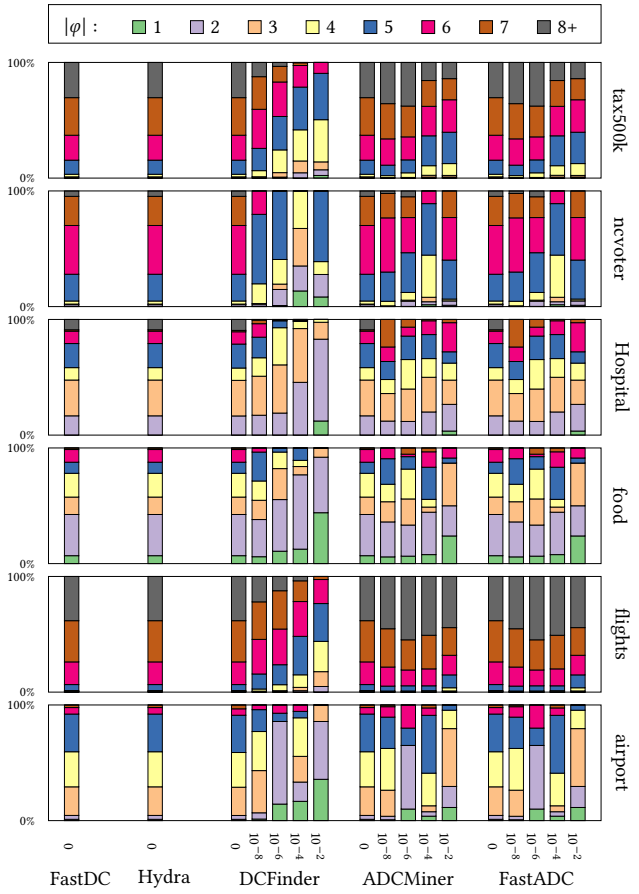

Figure 1: Number of predicates from DCs discovered by each algorithm (column) on each dataset (row) and with the given approximation factor ϵ (inner column), showing the large complexity of most discovered constraints in addition to the inability of approximation in reliably decreasing it.

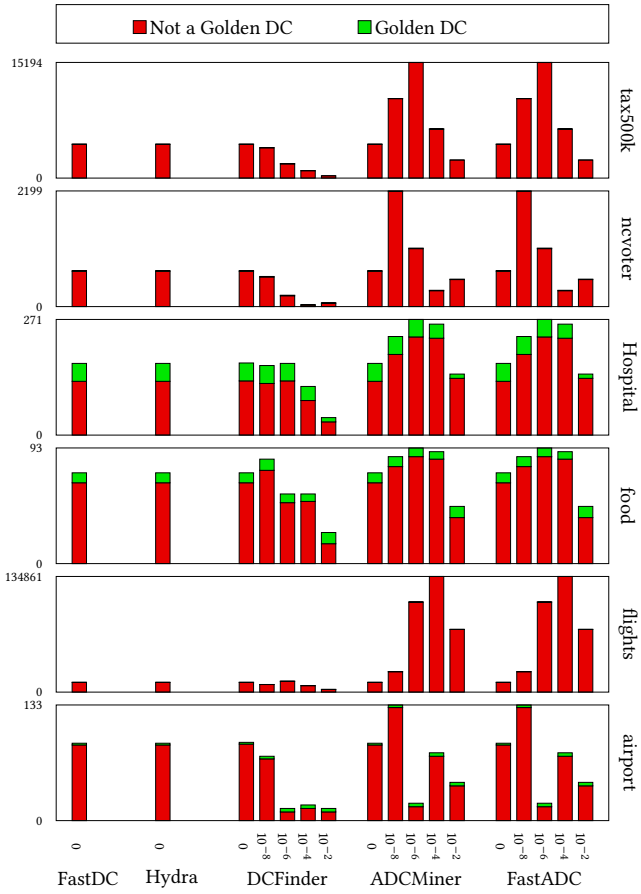

Figure 2: Number of discovered DCs that are golden, showing how most discovered DCs are not true constraints.

as several datasets do not substantially reduce their proportion of overly complex DCs after increasing approximation.

Instead, we can see how the use of approximation can sometimes exacerbate this issue, leading to larger and more complex DCs despite the attempts at making them more general. This phenomenon can be seen on the *ncvoter* dataset, where increasing the approximation beyond 0.0001 to achieve more general DCs paradoxically leads to the discovery of longer constraints. This behavior can be understood by simplifying the scenario. Let us have independent predicates P_1, P_2, P_3, with the same individual probability of fulfilling a random tuple pair $p(P_i) = 0.1$. If we look for minimal sets of predicates satisfied together for a proportion of tuple pairs below $\epsilon = 0.001$, only $\{P_1, P_2, P_3\}$ will be discovered. If we increase the approximation factor to $\epsilon = 0.01$, now three sets of predicates are discovered: $\{P_1, P_2\}$, $\{P_1, P_3\}$, $\{P_2, P_3\}$, showing how approximation can increase the number of discovered DCs. If we further increase approximation to $\epsilon = 0.1$, individual predicates are accepted, often leading to the discovery of less results (if $\{P_1\}$ is accepted, $\{P_1, P_2\}$, $\{P_1, P_3\}$ are no longer minimal). These behaviors can be combined, leading to approximation decreasing the number of short DCs while increasing the number of long DCs.

This result does not refute the claim that approximation can reduce the complexity of discovered DCs. However, it demonstrates that approximation alone is not always reliable for ensuring more general DCs. It also shows that approximation should not be considered a universal solution to the problem of overfitting.

The main goal of a discovery algorithm is not to discover small and specific constraints, but true ones representing real relationships of the data. The fact that approximation sometimes leads to the discovery of shorter DCs does not imply these are true data rules. In order to experimentally evaluate how good the algorithms are at discovering true DCs, some previous work used a set of golden DCs gathered by domain experts and evaluated how well their algorithms find those [8, 17, 23]. These publications present high recalls, highlighting how good their algorithms are at ensuring golden DCs are within the set of discovered constraints. However, they are generating a number of DCs that is several orders of magnitude larger than the amount of golden DCs.

We believe that analyzing the precision of the algorithms is crucial, as we must ensure the DCs discovered by the algorithms are actual constraints of the data. While prior work by [8] and [23]

initiated such an investigation, their focus on the top-k DCs based on specific metrics limits the scope of the analysis. By selecting only the top-k shortest constraints, they discover several high-quality DCs, but this cannot be generalized to broader results. Moreover, this approach is highly susceptible to underfitting when even minimal approximation is introduced.

We now present a comprehensive analysis of the precision yielded by the same DC discovery algorithms. We have generated a set of golden DCs for each dataset, as done previously in the literature [8, 17], through a combination of various techniques[3]:

- Golden DCs published in previous research [8, 17].
- Manual generation of simple constraints (Keys, FDs, equivalent columns), verified by domain experts.
- Manual generation of complex constraints by means of a thorough analysis by the domain experts.

We used this set of golden DCs to measure the precision of the algorithms, that is, which proportion of the discovered DCs are golden. This measure is a great indicator of the trust or certainty a user would have in obtaining true DCs from each algorithm. Figure 2 summarizes these experimental results, clearly showing how only a minuscule proportion of the discovered DCs correspond to true data constraints. Furthermore, this precision does not grow as the approximation increases, but it diminishes instead.

To sum up, while the literature considers approximation necessary to handle erroneous data and uncover more general DCs, this approach proves unreliable. It sometimes results in longer DCs and increases the number of false DCs discovered. Moreover, approximation introduces underfitting, causing algorithms to identify subsets of true rules as valid DCs, which then leads to the rejection of these true DCs due to minimality.

4 MODELING FALSE DCS

In this section, we analyze how the current DC Validity definition admits sets of completely independent predicates as valid DCs. While complete independence is rarely found in real data, this theoretical analysis allows us to model the appearance of overfit and underfit constraints in the results of all algorithms. We experimentally demonstrate how most results yielded by current DC discovery algorithms behave as sets of independent predicates.

The evaluation of a predicate P on a random pair of tuples can be treated as a binary random event, which can be modeled with a Bernoulli distribution with probability $p(P)$. This can be generalized to the joint evaluation of a set of k predicates $\varphi = \{P_1, \ldots, P_k\}$, where $p(\varphi) = p(\bigwedge_{i=1}^{k} P_i)$ is the probability of all predicates in φ being true. If the predicates in φ are independent, the probability of them jointly satisfying a random tuple pair equals the product of individual predicate probabilities $e(\varphi) = \prod_{i=1}^{k} p(P_i)$. Consider a repetition of this experiment for m random pairs of tuples, and let $m'(\varphi)$ be the number of tuple pairs fulfilling all predicates so that:

$$m'(\varphi) \sim Binomial(m, e(\varphi))$$

To accept φ as a valid DC, the proportion of tuple pairs that satisfy all the predicates (i.e. violating the DC) must be below the approximation factor ϵ (which is 0 for exact discovery). Thus, we define $p(m'(\varphi) \leq \epsilon \cdot m)$ as the probability of φ being accepted as valid. That is, the probability of the number of pairs of tuples violating the DC being lesser or equal than $\epsilon \cdot m$. This implies that any set of independent predicates, no matter how meaningless they read together, has a probability of being accepted as a true DC in our data. Ideally, false constraints would have very small acceptance probabilities, but in the following subsections we discuss that this is not the case, and all types of erroneous DCs presented in the literature can be explained due to sets of independent predicates.

[3]The set of golden DCs is available in the paper repository.

4.1 Underfit

Underfit DCs are generally formed by one or two very selective predicates that are accepted as a valid DC due to approximation. Let us consider the real examples of accepted underfit constraints on the *Tax* dataset shown in Table 2 (DCs $\varphi_{(2-5)}$). Table 4a lists the probabilities of each predicate involved being true for any random pair of tuples, or equivalently, the proportion of tuple pairs that satisfy each predicate. The first three predicates are true for a very small proportion of tuple pairs, as the large number of unique elements in the respective columns (*ZIP*, *Last Name*, *Area Code*) makes it unlikely to pick two random tuples sharing the same value. This means that when evaluating these predicates on a large enough number of tuple pairs (e.g. $m > 1000$), the probability of them being satisfied on less than $\epsilon = 1\%$ of the tuple pairs (hence being discovered as valid DCs) is essentially 100%, as shown in the first three columns of Table 4b.

Therefore, by just looking at the individual predicate probabilities, we can assert it is almost impossible to find enough tuple pairs that fulfill the predicates to reject them as uniqueness constraints. The same is not true for the false DC $\neg(t_x.ST = t_y.ST)$ (i.e., unique *State*), as shown in Table 4b. This makes sense as this predicate is expected to be true for around 1.9% of tuple pairs, so the chances of it being fulfilled less than 1% are negligible for a large enough number of tuple pairs. However, when a new independent predicate is added to this DC, the probability of both predicates being true at the same time drops to 0.9%, meaning it is quite likely for a set of $m > 1000$ tuple pairs to contain less than $\epsilon = 1\%$ fulfilling both. The last column of Table 4b showcases how just adding an independent predicate to $\neg(t_x.ST = t_y.ST)$ makes the chances of accepting this underfit constraint as a valid approximate DC go from practically impossible to almost certain.

For this reason, underfit DCs are essentially a small set of highly selective, independent predicates with a joint probability of being true together lower than the approximation factor. Clearly, combining independent predicates on unrelated attributes does not create any useful constraint, yet these DCs are still considered valid under the current DC validity definition.

4.2 Overfit

Overfit DCs are discovered due to the same principle discussed in the previous subsection. However, unlike underfit DCs that require using approximation to be discovered, overfit DCs also appear even when using no approximation ($\epsilon = 0$). Overfit DCs are typically characterized by a long list of seemingly unrelated predicates that are nevertheless accepted as a valid DC, such as φ_6 from Table 2. The individual probabilities of each predicate being fulfilled in a

Table 4: Mechanism behind the generation of underfit constraints. Selective predicates with probabilities lower than the approximation factor are accepted as valid DCs. If their individual probabilities are not sufficiently low, independent predicates can still be combined to form DCs with a joint probability that is low enough to be considered valid.

(a) Individual probabilities of predicates involved in underfit constraints: $\varphi_2, \varphi_3, \varphi_4, \varphi_5$.

P	$t_x.Zip = t_y.Zip$	$t_x.LN = t_y.LN$	$t_x.AC = t_y.AC$	$t_x.ST = t_y.ST$	$t_x.GD \neq t_y.GD$
$p(P)$	$4.5 \cdot 10^{-5}$	$9.9 \cdot 10^{-5}$	$8.8 \cdot 10^{-3}$	0.019	0.5

(b) Probability of sets of predicates being discovered as underfit DCs with approx. factor $\epsilon = 1\%$ on a dataset with $m > 1000$ tuple pairs.

φ	$\neg(t_x.Zip = t_y.Zip)$	$\neg(t_x.LN = t_y.LN)$	$\neg(t_x.AC = t_y.AC)$	$\neg(t_x.ST = t_y.ST)$	$\neg(t_x.ST = t_y.ST \wedge t_x.GD \neq t_y.GD)$
$e(\varphi)$	$4.5 \cdot 10^{-5}$	$9.9 \cdot 10^{-5}$	$8.8 \cdot 10^{-3}$	0.019	0.009
$p(m'(\varphi) < 0.01 \cdot m)$	~ 1	~ 1	~ 1	~ 0	~ 1

Table 5: Mechanism behind the generation of overfit constraints on clean data. Groups of predicates are agglomerated regardless of their relationships, with the only goal of making even a single pair of tuples in the data unlikely to fulfill them all at the same time.

(a) Individual probabilities of predicates involved in overfit constraint: φ_6.

P	$t_x.GD = t_y.GD$	$t_x.PH = t_y.PH$	$t_x.CT = t_y.CT$	$t_x.MS = t_y.MS$	$t_x.CH \neq t_y.CH$	$t_x.TR \neq t_y.TR$	$t_x.SX = t_y.SX$	$t_x.CX \leq t_y.CX$
$p(P)$	0.5	$2.5 \cdot 10^{-4}$	$7.2 \cdot 10^{-4}$	0.5	0.49	0.93	0.41	0.58

(b) Probability of there being no tuple pair fulfilling all predicates, among the $m = \frac{15000(15000-1)}{2}$ tuple pairs available to the algorithms that discovered this overfit DC.

φ	$\neg(t_x.GD = t_y.GD \wedge t_x.PH = t_y.PH \wedge t_x.CT = t_y.CT \wedge t_x.MS = t_y.MS \wedge t_x.CH \neq t_y.CH \wedge t_x.TR \neq t_y.TR \wedge t_x.SX = t_y.SX \wedge t_x.CX \leq t_y.CX)$
$e(\varphi)$	$5.9 \cdot 10^{-9}$
$p(m'(\varphi) = 0)$	0.51

random tuple pair are shown in Table 5a. Assuming these predicates are independent, Table 5b estimates the proportion of tuple pairs that will satisfy every predicate in φ_6 to be $5.9 \cdot 10^{-9}$, making the probability of no tuple pair fulfilling all predicates to be 51%. This probability is of little importance in practice, as new predicates can be added to bring this value as high as desired. The way these overfit DCs are constructed by current algorithms is by first obtaining a set of very selective predicates (such as predicates 2 and 3 in Table 5a), which are randomly fulfilled together by a small amount of tuple pairs. Then, for each of these tuple pairs, a random predicate not fulfilled by these tuples is added to the DC. This process is repeated until the target satisfaction is met.

Real overfit DCs often contain related subsets of predicates. For instance, there is a small correlation between the attributes *MaritalStatus (MS)* and *ChildrenStatus (CH)* in φ_6. Thus, their real joint probability slightly deviates from the expected under independence, but it does not change the fact that we can still add other independent predicates and obtain an accepted DC regardless of approximation. We further discuss this phenomenon in Section 5.

Overfit DCs, in essence, are generated by the same principle as underfit DCs: i.e. they agglomerate sets of independent predicates. While underfit DCs are accepted due to approximation, overfit DCs are accepted due to randomly generated sets of independent predicates that no tuple pair satisfies.

4.3 Analysis of current algorithms

We showed how the acceptance of overfit and underfit constraints can be attributed to the fact that state-of-the-art algorithms generate DCs by concatenating independent predicates, until their satisfaction is high enough. This indicates that, internally, the algorithms are generating sets of independent predicates as DCs, instead of discovering real constraints on the data.

To experimentally demonstrate that current algorithms often generate DCs without considering the underlying data relationships, we compare $p(\varphi)$ and $e(\varphi)$ for each discovered DC φ, as well as analyze how $e(\varphi)$ evolves as the number of tuples given to the algorithms increases.

Comparing $p(\varphi)$ and $e(\varphi)$: If the algorithms are generating DCs by agglomerating independent predicates, the joint probabilities of the predicate sets $p(\varphi)$ will be very similar to what is expected under independence $e(\varphi)$. We compute the proportion of discovered DCs whose true joint probability is low enough so that it would be unlikely under the assumption of independence with the following hypothesis test:

$$H_0 : p(\varphi) = e(\varphi)$$
$$H_1 : p(\varphi) < e(\varphi)$$

Under the null hypothesis, the number of tuple pairs fulfilling all predicates $m'(\varphi)$ follows a binomial distribution with probability $e(\varphi)$. If an algorithm operates with n tuples, there are $m = \frac{n(n-1)}{2}$ distinct pairs of them available. If k tuple pairs fulfill all predicates, the p-value of the one-sided test is $p(m'(\varphi) \leq k)$.

Figure 3 shows the proportion of discovered DCs for which this probability is lower than $\alpha = 5\%$. This graph showcases how, for a significant portion of the discovered DCs, we are unable to find

evidence against them being independent sets of predicates instead of true relationships.

At this point, we emphasize that predicate independence is not always fully correlated with DC quality for two reasons. First, an overfit DC may contain two slightly correlated predicates that lead to the rejection of the statistical test. This is the reason behind the low proportion of completely independent predicate sets in some datasets, as algorithms gather two or three negative correlated predicates, and pad them with several independent ones until satisfaction is high enough.

Second, a true DC may have negative and positive correlations between its predicates, leading to joint probabilities being what is expected under independence. This is a very unlikely case though, as predicates confirming true exact DCs have 0 joint probability, which cannot be obtained from the product of nonzero predicate probabilities. And for an approximate DC to have this shape, the positive correlations must be high enough to offset the negative correlations, but this will easily lead to nonminimal DCs as positively correlated predicates can be removed.

The goal of this experiment is not to determine DC quality, but to showcase how algorithms are indeed accepting sets of completely independent predicates as valid DCs. This concept of using statistically significant predicate relationships to determine which DCs correspond to true rules of the data is expanded in Section 5.

Evolution of $e(\varphi)$ as data increases: An alternative approach to prove how algorithms are just gathering sets of independent predicates can be taken by analyzing how the joint probability under independence $e(\varphi)$ evolves as we tweak the amount of data available to the algorithms. If the algorithms discover only true constraints, we would find that the values of $e(\varphi)$ do not change significantly as we increase the data available to the algorithms. This is due to the fact that once enough data is available to discover a true constraint φ, this one will always be discovered no matter how many more data are added. Any further executions with more data will still discover this very DC with the exact same $e(\varphi)$. However, if the algorithms were simply agglomerating independent predicates, the satisfaction of the discovered DCs would depend only on $e(\varphi)$. We know that with no approximation and n tuples, the probability of accepting a DC made of independent predicates is the probability of the predicates not being randomly true at the same time for all $m = \frac{n(n-1)}{2}$ tuple pairs, as: $p(m'(\varphi) \leq 0) = (1 - e(\varphi))^m$.

Hence, the sets of independent predicates need to have joint probabilities below $\frac{-log(\alpha)}{m}$ to maintain a reasonable acceptance probability above some α.

Proof: Assume the opposite: $e(\varphi) > \frac{-log(\alpha)}{m}$. This implies that

$$(1 - e(\varphi))^m < (1 + \frac{log(\alpha)}{m})^m$$

The LHS is the probability of accepting the set of predicates, and the RHS is bounded above by $e^{log(\alpha)} = \alpha$. Together, we have:

$$(1 - e(\varphi))^m < \alpha$$

□

As the data available to the algorithms grow, the joint probability of discovered sets of independent predicates needs to decrease to maintain a high probability of never being true at the same time for all available tuple pairs. Figure 4 shows how $e(\varphi)$ tends to always be proportional to $\frac{1}{n(n-1)}$, indicating that the discovered DCs are not true constraints of the data. Instead, this suggests that they are sets of independent predicates grouped together with the objective of making them unlikely to be true at the same time, thus ensuring a high acceptance probability.

We showed this experiment only for FastADC on the *Tax* dataset due to the computational cost of discovery algorithms, but the results are generalizable to all of them: all algorithms use the same DC validity definition, and even if individual implementations differ, none of them considers the source of satisfaction of a DC when classifying it as valid. Instead, to generate DCs, algorithms concatenate unrelated (independent) predicates until their joint satisfaction is high enough.

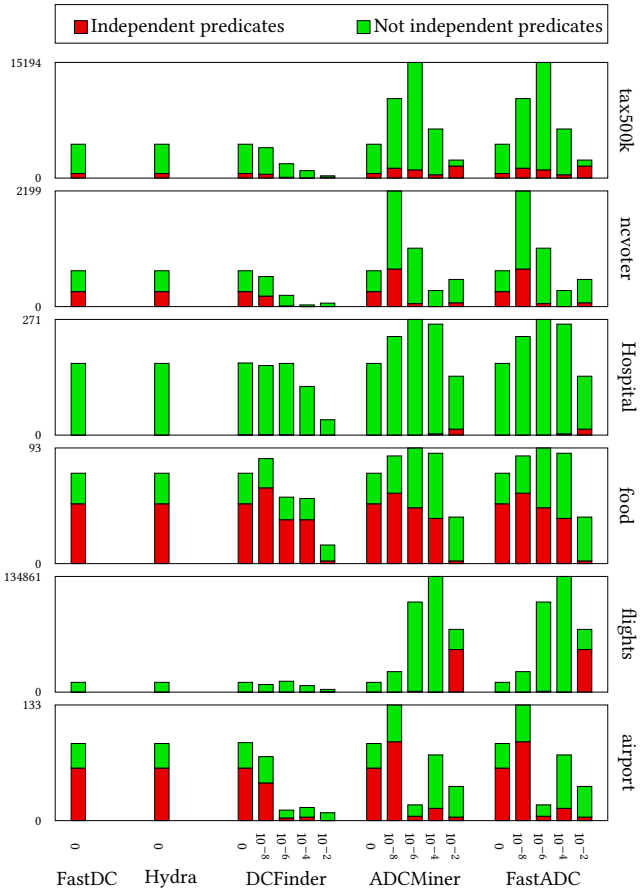

Figure 3: Number of discovered DCs whose predicate sets φ are true together on a number of tuples $m'(\varphi)$ unlikely ($\alpha = 0.05$) under independence $p(\varphi) = e(\varphi)$. It shows how many of the erroneous DCs from Figure 2 can be attributed to algorithms generating sets of independent predicates.

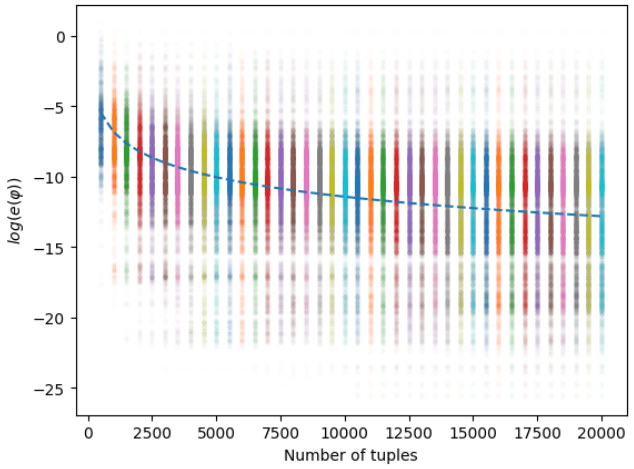

Figure 4: Evolution of the joint probability under independence $e(\varphi)$ of exact DCs discovered by FastADC on the Tax dataset as we increase the number of tuples given to the algorithm n. The dotted line follows the trend marked by $\frac{1}{n(n-1)}$. Since $e(\varphi)$ decreases at the same rate, the algorithm is simply agglomerating independent predicates.

5 REDEFINING DC VALIDITY

In this section we provide modifications to the DC validity definition to ensure that the discovered DCs always correspond to true relationships between the data attributes. To this end, we incorporate a new soundness property, and redefine the triviality property already present in the original definition.

5.1 Definition of Soundness

Soundness guarantees that the algorithms do not accept DCs whose satisfaction is achieved by agglomerating independent predicates. Instead, it ensures that only constraints representing real relationships between the predicates are discovered:

- **Soundness**: The satisfaction of the DC is not artificially achieved through independent predicates.

Given our analysis on sets of independent predicates, accepting only those constraints whose predicates are provably not independent seems the most intuitive option. However, comparing Figures 2 and 3 we can see how not all erroneous discovered DCs are sets of completely independent predicates. While rejecting these sets of independent predicates would greatly improve the quality of current results, the soundness rule should be general enough to ensure that valid DCs are completely free of any DC whose satisfaction is owed to independent predicates in any form.

Examples of erroneous DCs not completely made of independent predicates are well-known in the literature. We studied overfit DCs in the context of error-free data in Section 4, but overfit DCs are also formed when independent predicates are added to true DCs due to erroneous data. Consider the following exact DC, which is accepted on the *Tax* dataset when some errors are introduced in the data: $\neg(t_x.CT = t_y.CT \wedge t_x.ST \neq t_y.ST \wedge t_x.FN = t_y.FN)$ (Equal *City*, different *State*, equal *First Name*) . Clearly, this is an overfit DC, as the last predicate has only been added in order to ignore the tuple pairs affected by errors, yet this is not a set of independent predicates. The first two predicates correspond to a true constraint (FD: *City* $\rightarrow$ *State*) and are negatively correlated, while only the last one is unrelated to the rest.

The soundness rule should not simply prevent the acceptance of sets of completely independent predicates. Its goal is to reject any DC discovered by artificially adding independent predicates to increase satisfaction. We propose ensuring DC soundness through atomicity:

- **Atomicity**: Consider a set of predicates $\varphi = \{P_1, P_2, \ldots, P_k\}$, and let $p(P|\ \varphi \setminus \{P\})$ be the probability of predicate $P \in \varphi$ when all other predicates in φ are true. The predicate set φ is not atomic if it can be reduced $\varphi' \subset \varphi$ without changing the behavior of any remaining predicate $P \in \varphi'$:

$$p(P|\ \varphi \setminus \{P\}) = p(P|\ \varphi' \setminus \{P\}) \quad \forall \varphi' \subset \varphi, \forall P \in \varphi'$$

We use a statistical approach to determine whether this equality is fulfilled for each predicate. We assume the evaluation of a predicate X conditioned to $Y, Z \ldots$ follows a Bernoulli distribution with unknown, Beta-distributed, population probability $p(X|Y, Z \ldots)$. If there are a successes and b failures of predicate X when $Y, Z \ldots$ are true, we can update our knowledge about the probability distribution through Bayesian Learning: $p(X|Y, Z \ldots) \sim Beta(a+1, b+1)$. By modeling the log odds ratio between the two probabilities in each atomicity equality as done in [14] we can compute the probability of the LHS not being smaller than the RHS. The equality is not fulfilled if this probability is below some constant α[4]. If all equalities are proven to be unfulfilled with enough statistical significance, the DC is atomic.

If the set of predicates of a DC is atomic, no subset of them can be removed without altering the behavior of some predicate. This ensures the DC does not contain independent predicates and, instead, entirely owes its satisfaction to capturing a relationship between the predicates. For instance, DC φ_5 would be rejected under this new DC validity rule. This is because the distribution of the predicate $t_x.ST = t_y.ST$ does not change significantly when conditioned by the predicate $t_x.GD \neq t_y.GD$.

This way, we extend the findings from Section 4 to ensure that every discovered DC captures a meaningful relationship between predicates.

5.2 Redefinition of Triviality

While the soundness rule addresses all erroneous DCs discussed, there is some overlap between sets of independent predicates and trivial DCs that warrants discussion. The general definition of a trivial DC, as stated in [8], is: *"A DC is trivial if it is always satisfied by any data"*. This makes sense because the purpose of a constraint is to ensure that invalid data does not satisfy it. A DC that can never be false, no matter how erroneous the data, is useless. However, as this definition is too abstract to be validated in practice during the

[4]This parameter alters the sensitivity of the test. A low threshold increases false negatives by requiring lots of data to obtain enough statistical evidence, and a high threshold increases false positives by allowing randomness to be mistakenly treated as correlation. However, during our experiments, probabilities for uncorrelated predicates never fell below 0.01, meaning using any reasonable α value would lead to similar results.

discovery process, the literature relies on an alternative definition: *"A DC is trivial if one predicate's opposite is implied by other predicates."* This definition is much easier to validate, typically by checking DCs with multiple predicates over the same attribute.

Despite the vagueness of the general definition, there is some intersection between these trivial DCs and sets of independent predicates. Let us assume we have a database with k attribute columns A_i, and consider a DC formed by every possible equality predicate $\varphi = \neg(t_x.A_1 = t_y.A_1 \wedge t_x.A_2 = t_y.A_2 \wedge \cdots \wedge t_x.A_k = t_y.A_k)$. For the vast majority of datasets, this DC is valid due to the very low probability of finding two identical tuples. Even if one is willing to consider this a true constraint of the data, changing the operator of a subset of the predicates to any other, like $<$ or $\neq$, will still lead to a set of predicates that are never true at the same time. The number of satisfied DCs we can generate this way grows exponentially with the amount of attributes.

This kind of DCs clearly qualify as overfit, yet they would also fall within the general triviality definition. In the same way that DCs like $\neg(t_x.A_1 = t_y.A_1 \wedge t_x.A_1 \neq t_y.A_1)$ are always satisfied despite not representing a data constraint, long overfit DCs are also always satisfied without their predicates having any relationship. We identify two distinct kinds of triviality, characterized by the reason the DCs are always satisfied:

- **Syntactic Triviality:** These DCs are always satisfied due to the structure of the DC. In other words, the triviality of these DCs is known **before** even looking at the data. For example $\neg(P \wedge \neg P)$ will always be satisfied regardless of the predicate P or the values in the database.
- **Semantic Triviality**: These DCs are always satisfied due to the extremely low likelihood of all predicates being true at the same time. In other words, their triviality can only be identified **after** examining the data and estimating the probabilities of individual predicates. For example, it is impossible to estimate the satisfaction of φ_6 before looking at the data. However, once individual predicate probabilities are known, the probability of all of them being true together is minuscule, qualifying the DC as trivial.

In this case, semantically trivial DCs are a subset of non-sound DCs, as their satisfaction is owed to independent predicates. However, syntactically trivial DCs seem to correspond to an entirely different class than all other possible DCs, not just erroneous ones. To clarify this claim, we need to revisit the definition of Integrity Rules. According to [9, 28], Integrity Rules are defined, as: *"A mechanism for limiting the possible states of the database"*. This generally makes sense for DCs, as they prevent the data from being in states that violate certain rules, like Keys or Functional Dependencies [2, 31, 32]. The only exception are syntactically trivial DCs, as they prevent states that could never exist in the first place.

For this reason, along with the fact that our novel soundness rule already rejects semantically trivial DCs, we propose redefining DC triviality to eliminate redundancy among rules and to better distinguish this class of syntactically trivial constraints from others:

- **Nontriviality**: The DC does not restrict impossible states of the data.

With this redefinition, there is no overlap between soundness and triviality. The redefined triviality rule now explicitly prevents

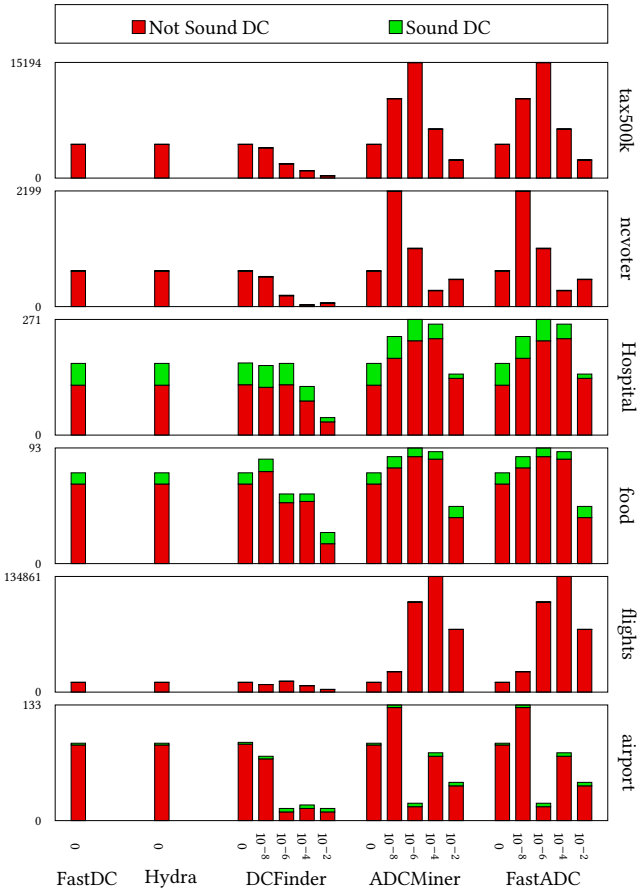

Figure 5: Number of discovered DCs that are sound, showing how the proportion of not sound DCs is similar to the proportion of false DCs from Figure 2.

syntactically trivial DCs like $\neg(t_x.ST = t_y.ST \wedge t_x.ST \neq t_y.ST)$ from being valid, which all algorithms already detect and reject. Our key contribution is the rejection of semantically trivial DCs like φ_6 for the first time thanks to the soundness rule. This allows our DC validity definition to ensure valid DCs are not "fulfilled for any data", as was the goal in the original formalisation of the DC discovery problem ([8]).

6 EXPERIMENTAL EVALUATION

In this section we evaluate the impact of our proposed definitions on the results produced by discovery algorithms. First, we compute the proportion of the discovered DCs that are not sound, showing how the number of sound DCs closely matches the number of golden DCs. Second, we analyze the DCs that meet our redefined DC validity criteria to confirm that sound DCs indeed correspond to true DCs. This analysis shows how incorporating the soundness rule significantly improves the quality of the results.

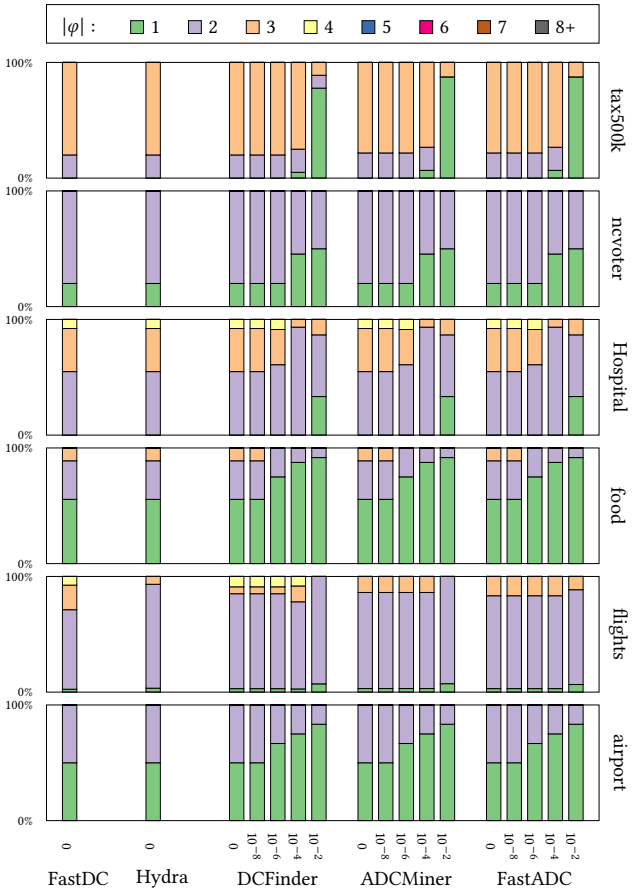

Figure 6: Number of predicates of resulting sound DCs. Shows how sound DCs have consistently reasonable lengths, unlike those shown in Figure 1.

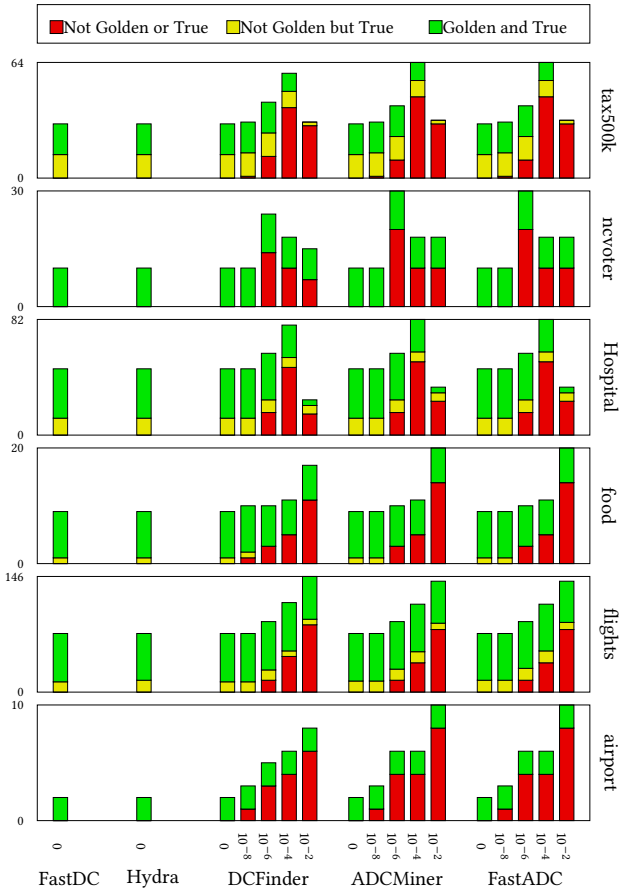

Figure 7: Number of discovered sound DCs that are golden and those that are true despite not being golden. Shows how sound DCs almost entirely correspond to true data constraints, unlike those shown in Figure 2.

6.1 Evaluation of DC Soundness

Now we study which proportion of the discovered DCs are sound, in order to assess how well our redefined DC validity could aid in reducing the amount of discovered DCs. Since the soundness rule is designed to prevent DCs containing independent predicates, the proportion of DCs it rejects should be no less than in Figure 3. However, if this new rule can detect more complex types of erroneous DCs, the proportion of false DCs detected could be as high as those in Figure 2, completely removing all erroneous DCs from the results.

To evaluate soundness, we run the algorithms for each dataset and different approximation factors, following the idea in Figures 2 and 3, and we checked soundness for the outputs produced by each run. Specifically, in order to estimate which proportion of the results are sound, we have determined the soundness of each discovered DC by checking if it fulfills the atomicity rule. Figure 5 shows how the soundness rule allows for the detection of a large number of erroneous DCs. The soundness rule not only properly discerns all sets of independent predicates, but it also detects large amounts of erroneous DCs that are more complex than simple sets of mutually independent predicates. These results are encouraging, because they imply that the soundness rule allows for a massive reduction in the number of discovered DCs. Most importantly, this reduction is achieved through an intuitive rule which simply aims to discard those DCs that the algorithms constructed through the agglomeration of independent predicates, keeping only those with statistically significant predicate relationships.

6.2 Analysis of Sound DC quality

As discussed in Section 3, simply reducing the number of DCs discovered does not guarantee they will be true. Here, we show how our redefined definition of DC validity can reliably lead to exclusively true constraints. To this end, we study the effect of our soundness rule in the following two dimensions:

- **Recall**: would our more restrictive DC validity increase the number of rejected true DCs?
- **Precision**: would our more restrictive DC validity decrease the number of accepted false DCs?

Table 6: Denial Constraints missed by domain experts but found with our redefined DC Validity on the Tax dataset.

DC	Semantics
$\neg(t_x.MX > t_y.MX \wedge t_x.SX > t_y.SX)$ $\neg(t_x.MX < t_y.MX \wedge t_x.SX < t_y.SX)$	An individual cannot have higher marital exemption (MX) and higher single exemption (SX) than another.
$\neg(t_x.MS = t_y.MS \wedge t_x.MX \neq t_y.MX \wedge t_x.SX \neq t_y.SX)$	If marital (MX) and single exemptions (SX) differ, marital status (MS) differs too.
$\neg(t_x.ST = t_y.ST \wedge t_x.MX = t_y.MX \wedge t_x.SX \neq t_y.SX)$ $\neg(t_x.ST = t_y.ST \wedge t_x.SX = t_y.SX \wedge t_x.MX \neq t_y.MX)$	For individuals in the same state (ST), marital (MX) and single exemptions (SX) determine each other.

Recall: First, we highlight that no golden DCs discovered by the algorithms have failed to be proven sound. Equivalently, our method achieved perfect recall throughout the experiments. This is because when a domain expert identifies an exclusive set of predicates forming a DC, this relationship consistently emerges during atomicity testing. This does not mean it is impossible for a true DC to fail being proven sound, as this may happen when the amount of data available is too small for the atomicity tests to be statistically significant. However, this is an encouraging result, as it showcases how the increased restrictiveness of our DC validity definition does not lead to a loss in recall.

Precision: Next, we study the effect of soundness in the complexity and precision of the results by repeating the experiments discussed in Section 3.3, but only with the subset of discovered DCs that are sound. Figure 6 shows how sound DCs are much shorter compared to the general results of the algorithms (Figure 1). These results confirm our hypothesis that algorithms are generating long overfit DCs by adding independent predicates until satisfaction is high enough, while DCs that are sound (and thus represent existing relationships in the data) are short and concise.

As before, discovering shorter DCs is not a guarantee of increased quality. We now evaluate the precision of the sets of sound DCs and compare these results to those presented in Figure 2. Given that the soundness rule leads to a significant drop in the number of discovered DCs and in their complexity, we have been able to manually analyze all sound DCs to determine their truthfulness. All sound DCs have been classified according to two properties:

- **Golden:** is the DC golden? As done in Figure 2, we will analyze which proportion of sound DCs are golden.
- **True:** is the DC determined to be true by a domain expert? As done in [8], smaller sets of DCs can be analyzed manually by domain experts to more accurately compute the precision.

Figure 7 shows the proportion of discovered sound DCs that are part of the golden DC set. The results demonstrate a significant improvement in precision when our redefined DC validity is applied to determine which DCs to accept. Furthermore, several of the sound DCs that lie outside the set of golden DCs still appear to represent true data rules. Table 6 lists five examples of DCs satisfied in the *Tax* dataset that were not generated manually by domain experts. Not only these constraints are satisfied by the data, but they are also sound, indicating statistically significant relationships among predicates. This is strong evidence supporting our DC validity definition, as it enables the automatic discovery of new DCs that could not have been discovered only through domain knowledge.

In light of this, we have gathered all DCs of up to 4 predicates that are valid under our new DC validity definition. These DCs have been made publicly available in the paper repository, and highlight how adopting our redefined DC validity may allow algorithms to directly discover high-quality sets of DCs.

7 CONCLUSIONS

In this paper we presented a comprehensive experimental evaluation on the quality of the results given by several state-of-the-art DC discovery algorithms. This analysis highlights how current algorithms yield extremely long and contrived constraints, and how approximation fails in mitigating these issues. More importantly, we showed how the discovered DCs are rarely true constraints, meaning all of these algorithms are difficult to be applied in practice. We explained the discovery of all erroneous DCs discussed in the literature, namely underfit and overfit constraints, by understanding how independent predicates can be used to achieve satisfied DCs. Our findings suggest that the erroneous results reported in many publications of the field are not due to flaws in the specific algorithms employed, but rather stem from fundamental issues with the DC validity definition itself.

In response to these issues, we proposed a redefinition of DC validity to prevent the discovery of false DCs. We presented the soundness rule to prevent algorithms from generating DCs by agglomerating independent predicates, thus ensuring discovered DCs owe their satisfaction to representing some actual rule of the data. Finally, we validated the effect the soundness rule has in the quality of the discovered DCs. We showed how the number of erroneous DCs that are discovered is lowered by over 95% in most cases, with no decrease in recall. Furthermore, by just filtering results using our novel soundness rule, we have been able to discover constraints missed by domain experts, demonstrating how our redefined DC validity can be used to reliably discover true rules of the data.

ACKNOWLEDGMENTS

This work is supported by the Horizon Europe Programme under GA.101135513 (CyclOps) and the Spanish Ministerio de Ciencia e Innovación under project PID2020-117191RB-I00 / AEI/10.13039/501100011033 (DOGO4ML). Anna Queralt is a Serra-Húnter fellow. E. Almeida is funded by the CNPQ grants 302909/2022-2 and 444192/2024-7.

REFERENCES

[1] Naser Ayat, Hamideh Afsarmanesh, Reza Akbarinia, and Patrick Valduriez. 2012. Pay-As-You-Go Data Integration Using Functional Dependencies. In *Multidisciplinary Research and Practice for Information Systems*. 375–389.

[2] Laure Berti-Équille, Hazar Harmouch, Felix Naumann, Noël Novelli, and Saravanan Thirumuruganathan. 2018. Discovery of Genuine Functional Dependencies from Relational Data with Missing Values. *Proc. VLDB Endow.* 11, 8 (2018), 880–892. https://doi.org/10.14778/3204028.3204032

[3] Lingfeng Bian, Weidong Yang, Jingyi Xu, and Zijing Tan. 2024. Discovering Denial Constraints Based on Deep Reinforcement Learning. In *Proceedings of the*

33rd ACM International Conference on Information and Knowledge Management. 120–129.

[4] Johann Birnick, Thomas Bläsius, Tobias Friedrich, Felix Naumann, Thorsten Papenbrock, and Martin Schirneck. 2020. Hitting Set Enumeration with Partial Information for Unique Column Combination Discovery. *Proc. VLDB Endow.* 13, 11 (2020), 2270–2283. http://www.vldb.org/pvldb/vol13/p2270-birnick.pdf

[5] Tobias Bleifuß, Sebastian Kruse, and Felix Naumann. 2017. Efficient denial constraint discovery with hydra. *Proceedings of the VLDB Endowment* 11, 3 (2017), 311–323.

[6] Fei Chiang and Renée J Miller. 2008. Discovering data quality rules. *Proceedings of the VLDB Endowment* 1, 1 (2008), 1166–1177.

[7] Jan Chomicki, Jerzy Marcinkowski, and Slawomir Staworko. 2004. Computing consistent query answers using conflict hypergraphs. In *Proceedings of the 2004 ACM CIKM International Conference on Information and Knowledge Management, Washington, DC, USA, November 8-13, 2004*, David A. Grossman, Luis Gravano, ChengXiang Zhai, Otthein Herzog, and David A. Evans (Eds.). ACM, 417–426. https://doi.org/10.1145/1031171.1031254

[8] Xu Chu, Ihab F Ilyas, and Paolo Papotti. 2013. Discovering denial constraints. *Proceedings of the VLDB Endowment* 6, 13 (2013), 1498–1509.

[9] AnHai Doan, Alon Halevy, and Zachary Ives. 2012. *Principles of data integration*. Elsevier.

[10] Wenfei Fan, Floris Geerts, Jianzhong Li, and Ming Xiong. 2010. Discovering conditional functional dependencies. *IEEE Transactions on Knowledge and Data Engineering* 23, 5 (2010), 683–698.

[11] Stella Giannakopoulou, Manos Karpathiotakis, and Anastasia Ailamaki. 2020. Cleaning denial constraint violations through relaxation. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. 805–815.

[12] Arvid Heise, Jorge-Arnulfo Quiané-Ruiz, Ziawasch Abedjan, Anja Jentzsch, and Felix Naumann. 2013. Scalable Discovery of Unique Column Combinations. *Proc. VLDB Endow.* 7, 4 (2013), 301–312. https://doi.org/10.14778/2732240.2732248

[13] Yka Huhtala, Juha Kärkkäinen, Pasi Porkka, and Hannu Toivonen. 1999. TANE: An efficient algorithm for discovering functional and approximate dependencies. *The computer journal* 42, 2 (1999), 100–111.

[14] Norman L Johnson, Samuel Kotz, and Narayanaswamy Balakrishnan. 1995. *Continuous univariate distributions, volume 2*. Vol. 2. John wiley & sons.

[15] Sebastian Kruse and Felix Naumann. 2018. Efficient Discovery of Approximate Dependencies. *Proc. VLDB Endow.* 11, 7 (2018), 759–772. https://doi.org/10.14778/3192965.3192968

[16] Philipp Langer and Felix Naumann. 2016. Efficient order dependency detection. *VLDB J.* 25, 2 (2016), 223–241. https://doi.org/10.1007/S00778-015-0412-3

[17] Ester Livshits, Alireza Heidari, Ihab F. Ilyas, and Benny Kimelfeld. 2020. Approximate Denial Constraints. *Proc. VLDB Endow.* 13, 10 (2020), 1682–1695. https://doi.org/10.14778/3401960.3401966

[18] Thorsten Papenbrock and Felix Naumann. 2016. A Hybrid Approach to Functional Dependency Discovery. In *Proceedings of the 2016 International Conference on Management of Data, SIGMOD Conference 2016, San Francisco, CA, USA, June 26 - July 01, 2016*, Fatma Özcan, Georgia Koutrika, and Sam Madden (Eds.). ACM, 821–833. https://doi.org/10.1145/2882903.2915203

[19] Thorsten Papenbrock and Felix Naumann. 2017. Data-driven Schema Normalization. In *Proceedings of the 20th International Conference on Extending Database Technology, EDBT 2017, Venice, Italy, March 21-24, 2017*, Volker Markl, Salvatore Orlando, Bernhard Mitschang, Periklis Andritsos, Kai-Uwe Sattler, and Sebastian Breß (Eds.). OpenProceedings.org, 342–353. https://doi.org/10.5441/002/EDBT.2017.31

[20] Primal Pappachan, Shufan Zhang, Xi He, and Sharad Mehrotra. 2024. Preventing Inferences Through Data Dependencies on Sensitive Data. *IEEE Transactions on Knowledge and Data Engineering* 36, 10 (2024), 5308–5327. https://doi.org/10.1109/TKDE.2023.3336630

[21] Marcel Parciak, Sebastiaan Weytjens, Niel Hens, Frank Neven, Liesbet M. Peeters, and Stijn Vansummeren. 2024. Measuring Approximate Functional Dependencies: A Comparative Study. In *40th IEEE International Conference on Data Engineering, ICDE 2024, Utrecht, The Netherlands, May 13-16, 2024*. IEEE, 3505–3518. https://doi.org/10.1109/ICDE60146.2024.00270

[22] Eduardo HM Pena and Eduardo Cunha de Almeida. 2018. BFASTDC: A bitwise algorithm for mining denial constraints. In *Database and Expert Systems Applications: 29th International Conference, DEXA 2018, Regensburg, Germany, September 3–6, 2018, Proceedings, Part I 29*. Springer, 53–68.

[23] Eduardo HM Pena, Eduardo C De Almeida, and Felix Naumann. 2019. Discovery of approximate (and exact) denial constraints. *Proceedings of the VLDB Endowment* 13, 3 (2019), 266–278.

[24] Eduardo HM Pena, Fabio Porto, and Felix Naumann. 2022. Fast Algorithms for Denial Constraint Discovery. *Proceedings of the VLDB Endowment* 16, 4 (2022), 684–696.

[25] Eduardo H. M. Pena, Erik Falk, Jorge Augusto Meira, and Eduardo Cunha de Almeida. 2018. Mind Your Dependencies for Semantic Query Optimization. *J. Inf. Data Manag.* 9, 1 (2018), 3–19. https://sol.sbc.org.br/journals/index.php/jidm/article/view/1633

[26] Chaoqin Qian, Menglu Li, Zijing Tan, Ai Ran, and Shuai Ma. 2023. Incremental discovery of denial constraints. *The VLDB Journal* 32, 6 (2023), 1289–1313.

[27] Theodoros Rekatsinas, Xu Chu, Ihab F Ilyas, and Christopher Ré. 2017. Holoclean: Holistic data repairs with probabilistic inference. *arXiv preprint arXiv:1702.00820* (2017).

[28] Avi Silberschatz, Henry F. Korth, and S. Sudarshan. 2020. *Database System Concepts, Seventh Edition*. McGraw-Hill Book Company. https://www.db-book.com/

[29] Jaroslaw Szlichta, Parke Godfrey, Jarek Gryz, and Calisto Zuzarte. 2013. Expressiveness and Complexity of Order Dependencies. *Proc. VLDB Endow.* 6, 14 (2013), 1858–1869. https://doi.org/10.14778/2556549.2556568

[30] Nicolas Tamalu, Leandro Augusto Ensina, Eduardo Cunha de Almeida, Eduardo Henrique Monteiro Pena, and Luiz Eduardo Soares de Oliveira. 2023. Fault Detection in Transmission Lines: a Denial Constraint Approach. In *Proceedings of the 38th Brazilian Symposium on Databases, SBBD 2023, Belo Horizonte, MG, Brazil, September 25-29, 2023*. SBC, 231–243. https://sol.sbc.org.br/index.php/sbbd/article/view/25530

[31] Ziheng Wei and Sebastian Link. 2019. Discovery and Ranking of Functional Dependencies. In *35th IEEE International Conference on Data Engineering, ICDE 2019, Macao, China, April 8-11, 2019*. IEEE, 1526–1537. https://doi.org/10.1109/ICDE.2019.00137

[32] Ziheng Wei and Sebastian Link. 2023. Towards the efficient discovery of meaningful functional dependencies. *Inf. Syst.* 116 (2023), 102224. https://doi.org/10.1016/J.IS.2023.102224

[33] Renjie Xiao, Zijing Tan, Haojin Wang, and Shuai Ma. 2022. Fast approximate denial constraint discovery. *Proceedings of the VLDB Endowment* 16, 2 (2022), 269–281.

[34] Chen Ye, Haoyang Duan, Hua Zhang, Yifan Wu, and Guojun Dai. 2024. Learned Query Optimization by Constraint-Based Query Plan Augmentation. *Mathematics* 12, 19 (2024), 3102.