Journal of Zhejiang University-SCIENCE A (Applied Physics & Engineering) in press
www.jzus.zju.edu.cn; www.springer.com/journal/11582
E-mail: wthu@zju.edu.cn

Electronic Supplementary Materials

For https://doi.org/10.1631/jzus.A2200156

Complex integrity constraint discovery: measuring trust in modern intelligent railroad systems

Wen-tao HU, Da-wei JIANG, SaiWU, Ke CHEN, Gang CHEN[✉]
Key Lab of Intelligent Computing Based Big Data of Zhejiang Province, Zhejiang University, Hangzhou 310027, China
✉ Gang Chen, cg@zju.edu.cn

Data S1

1 Example

Table S1 The sample train operation dataset shows only the attributes of train departure time, next stop code, total mileage, traction energy consumption, and braking energy consumption in the dataset. The dataset does not contain the attribute (traction energy - braking energy)/total mileage, but the tuple (e.g., the last tuple) has a higher attribute value on this attribute than the other tuples, indicating that the current train has abnormal energy

	Time	Position (P)	Mileage (M)	Traction (T)	Braking (B)	(T-B) M
t_1	14:55	41	258	2160	200	7.59
t_2	15:30	41	279	2330	230	7.52
t_3	16:10	41	296	2460	240	7.50
t_4	17:40	41	342	2840	260	7.54
t_5	20:20	41	428	3640	340	7.71

Example: We train a linear regression model to predict whether a train is delayed using a dataset describing train operations, including train departure time, following stop code, total mileage, traction energy consumption, and braking energy consumption. This linear regression model is trained on a subset of this dataset that contains only records of train afternoon runs (e.g., the first four tuples in Table S1). In the evaluation of the accuracy of the regression model, we found that the average error of the regression model output for the evening runs (e.g., tuple t_5) differs by more than 5% compared to the afternoon runs of the trains. This is because the trains indicating evening runs deviate from the training data features that show only afternoon train runs. Specifically, trains running in the afternoon satisfy the following: "Trains with the same code at the next stop are very close in the energy consumption difference between traction and braking to their total mileage. Trains that travel at night do not support this constraint. It is worth noting that this constraint relies only on the existing attributes in the table and does not involve information on whether the marked trains are delayed or not. In addition, this constraint is not used in constructing the regression model, but it is a good reference standard to measure the performance of the regression model.

The sample example shows that when the training data contains interrelationships between numerical attributes (e.g., between train tractive force, train braking force, and operating mileage for predicting train operating speed), then the deployed model sets the potential interrelationships as invariant constraints. Complex integrity constraints can represent such invariant interrelationships and mark tuples (shuttles running at night) that violate such constraints.

In this example, we train a linear regression model to predict whether the train is currently likely to be late. t_1, t_2, t_3, t_4 model inference results in an expected train arrival, while these four tuples also meet the complex integrity constraints and are safe, and the model inference results are plausible. For t_5 tuple, the result of model inference is that the train arrives normally. However, t_5 violates the complex integrity constraint and is an unsafe tuple, and the inference result is not credible. If the user (driver) does not take additional acceleration operation according to the model inference result, it may cause the train to be late and thus cause some economic loss.

2 Related work

Table S2 summarizes previous work on related problems, but our scope is quite different. Specifically, we can detect data violations based on the complex integrity constraints found without the ground truth. This situation is critical in many practical applications of smart transportation when we detect extreme data violations. In this case, the current data to be verified do not immediately yield true results. For example, consider an urban subway autopilot scenario where the subway utilizes a superior trained controller to generate actions based on train speed, the relative distance between stations, and current load. In this case, we only need to alert the driver to take over control of the vehicle based on determining whether the sensor readings meet complex integrity constraints. The system guides the trustworthiness of the train operation model by detecting the level of insecurity of the data used in real-time through complex integrity constraints.

Complex integrity constraints belong to the category of data parsing, which refers to the task of extracting specific metadata in the dataset. Functional Dependencies (FD) (Y.Huhtala et al., 1999; Wu, P et al, 2020; Fan, W et al., 2020;Livshits, E., 2020; Kossmann et al, 2022) and their variants obtain the existence of a relationship between two sets of attributes without providing an expression in the form of parameters (W. Fan et al., 2010; Sebastian Kruse et al., 2018; L. Caruccio et al., 2016). Another more complex study, the Denial Constraint (DC) can contain many different constraints, such as FD and its variants (Tobias Bleifuß et al., 2017; Berti-Equille et al., 2018; Pena, E.H.,2021） However, this can make the constraints extremely complex and large, and it will be a challenge to sift through them to find potentially useful information. The goal of Pattern Functional Dependencies (PFD) is to address the deficiencies that exist in DC, but it targets attributes whose values are text. This is not applicable for intelligent rail systems where the majority of attributes are numeric. By using regular expressions and coding numbers as characters, constraints with different semantics are quickly detected (Pena, E.H et al., 2019 ; Qahtan et al., 2020; Tang, N.,2020).

The presence of noisy data in a dataset is commonly present. To reduce the impact of noise on the discovery of constraints, FDs and DCs relax the concept of constraints or add additional parameters to allow a portion of the data to violate the constraints (Xu Chu et al., 2013; Breve, B et a; .,2022). Embedded Uniqueness Constraints(eUCs) constraints represents unique column combinations embedded in complete fragments of incomplete data (Wei, Z et al., 2019; Link, S. et al 2019; Wei, Z. et al 2020). RFD considers similar characteristics of attribute values rather than equal characteristics(Caruccio et al., 2020a,2020b,2020c).In contrast, complex integrity constraints do not require any parameters from the user and discover complex integrity constraints in noisy datasets. Table S3 shows the information on the variables that appear in this paper.

3 Contribution

- We present the motivation for our work based on a Trusted Machine Learning (TML) case study.
- By studying the working definition of complex integrity constraints (Hu et al., 2020), we describe a consistent language for expressing complex integrity constraints in an intelligent railroad system.
- In modern intelligent railroad systems, some complex integrity constraints do not quantify the performance of models deployed in the system. To assess model performance by complex integrity constraints to advertise practical complex integrity constraints, we propose a concept of constraint importance and measure the constraints accordingly.
- To discover complex integrity constraints, we design a novel constraint discovery algorithm. The algorithm utilizes the BitVector indexing technique to vectorize the categorical attribute values in the database into one-hot encoded matrices so that the metric value of each attribute value can be quickly computed by transforming the query into a matrix bit operation without scanning the full table. We also analyze its running time and memory complexity.
- We empirically analyze the effectiveness of complex integrity constraints in a trust machine learning case study. We show that complex integrity constraints can reliably predict the trustworthiness of linear models, outperforming the state of the art.

Table S2: Complex Integrity Constraint complement existing constraint discovery and provide an efficient mechanism to quantify trust in prediction.

Legend FD: Functional Dependency √ : Applicable ⊥ : Not applicable	Coverage measure	Algebraic operation	Domain distribution
Complex Integrity Constraint (CIC)	√	√	√
Functional Dependency (FD)	⊥	⊥	⊥
Conditional FD	⊥	⊥	⊥
Denial Constraint (DC)	⊥	⊥	⊥
Pattern Functional Dependencies (PFD)	⊥	⊥	⊥
Embedded Uniqueness Constraints(eUCs)	√	⊥	⊥
Relaxed Functional Dependencies (RFDs)	√	⊥	⊥
Approximate FD	√	⊥	⊥

Table S3 Description of the Notation.

Notation	Domain	Description
m	$\mathbb{R}$	Number of attributes
n	$\mathbb{R}$	Number of tuples
$\mathcal{R}$	$\mathbb{R}^{\{m \times n\}}$	Relation schema
X_i	$\mathbb{R}^{\{1 \times n\}}$	i^{th} attribute of $\mathcal{R}$
Dom(X)	$\mathbb{R}$	Domain of attribute *X*
X_v	$\mathbb{R}$	Attribute values of the *X*
N_i	$\mathbb{R}^{\{1 \times n\}}$	Numerical attribute
$\oplus$	$\{+,-,\times,\div\}$	Algebraic constraint
ub, lb	$\mathbb{R}$	the upper and lower bounds
r	$\mathbb{R}^{\{m \times n\}}$	Instance defined over schema
Φ	-	Complex integrity constraint
t	$\mathbb{R}^{\{m \times 1\}}$	Tuple defined over instance
V_i	[0, 1]	i^{th} vector value of t
q	-	Query represented by the bit vector
A_v, B_v, C_v, D_v	$\mathbb{R}^{\{1 \times n\}}$	Attribute values A, B, C, D
S	$0 \le S \le 1$	Support threshold
C	-	Class of functions
$f,\quad g$	-	Function

3 Object and methods

3.1 Problem Definition

Trusted machine learning (TML) refers to the problem of quantifying the performance of an inference model deployed in an intelligent transportation system (Ghosh et al., 2016; Toreini, E et al., 2020; Qolomany, B et al .,2020). When an inference model is trained using a currently collected traffic dataset, the complex integrity constraints present in that dataset specify a safety range that describes the data on which the model is likely to make credible inferences. If the data exceed the safety range (violates the complex integrity constraint), then the deployed model may produce an untrustworthy result. Generally, the lower the level of violation, the lower the level of trust.

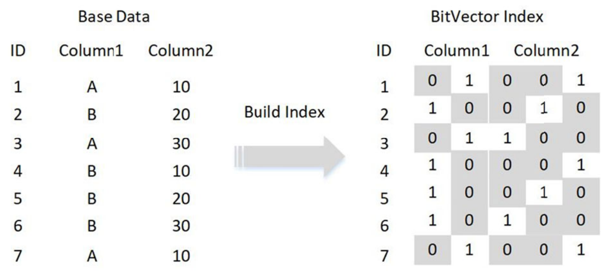
The TML problem is different from quantifying a dataset's credibility, and the difference is that the data assumptions are different. The problem of quantifying the credibility of a dataset assumes that the data may contain erroneous data and quantifies the credibility of the dataset by detecting the proportion of incorrect data. The TML problem is to divide the dataset into a training set and a dataset to be inferred. It assumes that the training set is reliable and finds unsafe data from the data to be inferred. In practical problems, the data to be inferred are often generated in real-time.

In the context of trusted machine learning, we formalize the concept of unsafe tuples on which predictions may be untrustworthy. We establish that complex integrity constraints provide a robust and complete procedure for detecting unsafe tuples, which suggests that complex integrity constraints should guide the models considered by intelligent transportation systems.

3.2 Index Construction

BitVectors. The categorical attribute vector is a bitmap over a tuple t of an instance r. The tuple t has d unique categorical attributes, hence tuple t contains d value bitvectors VB:$\{V_1, V_2, \ldots, V_d\}$.

The BitVectors Internals. For every value of the domain of the indexed attribute, BitVectors store a value vector showing which bits contain the corresponding value. For example, Fig. S1 shows the two categorical attribute vectors corresponding to the seven tuples that attribute 1 values A and B and attribute 2 values 10, 20, and 30, respectively. Rows 1, 3 and 7 are equal to A, and rows 2, 4, 5, and 6 are equal to B and rows 1, 4, and 7 are equal to 10, and rows 2, and 5 are equal to 20, and rows 3, and 6 are equal to 30. As a result, the value bitvectors have the corresponding bits set to one.



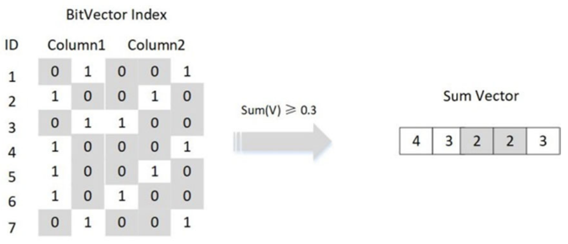
ID	Column1	Column2
1	A	10
2	B	20
3	A	30
4	B	10
5	B	20
6	B	30
7	A	10

ID	Column1		Column2		
1	0	1	0	0	1
2	1	0	0	1	0
3	0	1	1	0	0
4	1	0	0	0	1
5	1	0	0	1	0
6	1	0	1	0	0
7	0	1	0	0	1

Fig. S1 The internals of BitVector

Value-BitVector Mapping (VBM). A common operation in the BitVector index is to locate the bit corresponding to a specific value v of the domain attribute. The query that needs to search the bitvector index performs this action. Then, once we have indexed the instance r, we use the index to quickly calculate the support of each categorical attribute value, as well as to find those categorical attribute values that satisfy the minimum support and to calculate the μ and σ of the derived attributes. To do this, we have designed two operators for search and computation.

Support Calculation. We first discuss how BitVectior is probed for support calculation; that is, how we calculate the attribution support and find whether an attribution value meets the support threshold. The first step is to sum the bitvector i that corresponds to the times of domain attribution using the VBM which links values to bitvectors. The next step is to check whether the support of attribution value over support threshold that are marked on V_i . If all bits in V_i are unset then there is no attribution pruning. Otherwise, if even a single bit is set, then prunes the domain of attribution. For example, in Fig. S2 we probe for a support threshold equal to 0.3. In this case, we first sum the bitvectors for the support value and check whether the value exceeds the threshold.

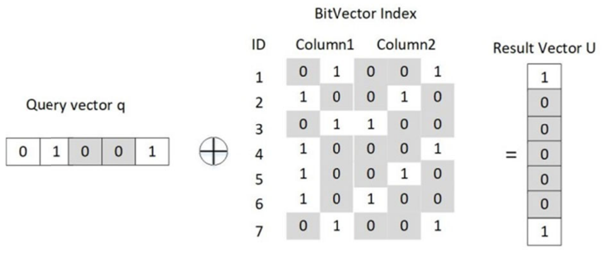


ID	Column1		Column2		
1	0	1	0	0	1
2	1	0	0	1	0
3	0	1	1	0	0
4	1	0	0	0	1
5	1	0	0	1	0
6	1	0	1	0	0
7	0	1	0	0	1

4	3	2	2	3

Fig. S2 Support calculation for the sum vector with bitvectors.

Search. We continue with how BitVector handles exact match queries, which means how we find whether a tuple t exists in the indexed column, and in which position. The first step is to construct a query represented by the bit vector q. The next step is to check whether BitVector bits equal to q with the AND operation that is marked. For example, in Fig. S3 we probe for a query which Column 1 equals to A and Columns 2 equals to 10. In this case we first construct the query vector q and then with the AND operation, we obtain the positions that contain this query q.

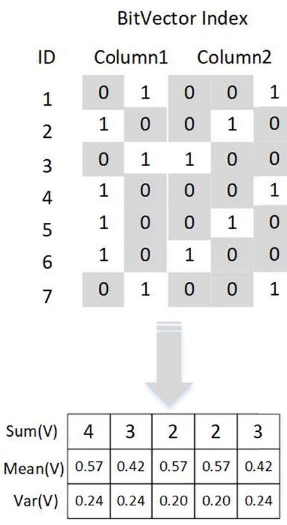


0	1	0	0	1

ID	Column1		Column2			Result Vector U
1	0	1	0	0	1	1
2	1	0	0	1	0	0
3	0	1	1	0	0	0
4	1	0	0	0	1	0
5	1	0	0	1	0	0
6	1	0	1	0	0	0
7	0	1	0	0	1	1

Fig. S3 Search for results with BitVector.

Mean and variance calculation. We then consider how BitVector handles the mean and variance calculation, i.e., how we calculate the normal range of a set of tuples in the derived numerical attribution. The first step is to construct the numerical attribution represented by the bit vector q. The next step is to perform sum, mean, and variance operations. For example, in Fig. S4 we probe for operations which sum Column 1 and column2. In this case, we first construct the query vector q and then with the AND operation, we obtain the positions that contain this query q.



ID	Column1		Column2		
1	0	1	0	0	1
2	1	0	0	1	0
3	0	1	1	0	0
4	1	0	0	0	1
5	1	0	0	1	0
6	1	0	1	0	0
7	0	1	0	0	1

Sum(V)	4	3	2	2	3
Mean(V)	0.57	0.42	0.57	0.57	0.42
Var(V)	0.24	0.24	0.20	0.20	0.24

Fig. S4 The mean and variance calculation for results with BitVector.

Time Complexity. The time complexity of the search operator and the computation operator is $O(n)$. The search operator runs $O(\mathrm{m})$ times in the first loop, where $O(\mathrm{m})$ is the number of attributes. The execution time is mainly related to the size of r, where r is a matrix of the $[m, \mathrm{n}]$ structure of the matrix, and n is much larger than n, where m is the number of tuples in instance r. The response time of the Sup computation operator is determined whose response time also depends on the number of tuples in instance r. Thus, the algorithmic response time for the search operator and the computation operator is $O(n)$.

3.3 Inference axioms for Complex Integrity Constraint

We developed an inference system for complex integrity constraints, similar to Armstrong's axioms in FDs (P. Bohannon et al., 2007). Since the main search space of complex integrity constraints lies in the selection of conditional values for categorical attributes, our inference system is mainly used for the optimization of this part.

Let A_v, B_v, C_v, D_v be the attribute values of the categorical attributes A, B, C, D in the relation instance r. In the instance r, if the value A_v of the tuple in attribute A can determine the value B_v in attribute B, then we say that A_v can determine B_v, notated as $A_v \rightarrow B_v$. For simplicity, we write A_vB_v to represent the set of two attribute values A_v and B_v instead of $A_v \cup B_v$.

Union Rule. If $A_v \rightarrow B_v$, $A_v \rightarrow C_v$, then $A_v \rightarrow B_vC_v$. This axiom states that an attribute value determines each of the other two attribute values; then, it must also determine the concatenation of the other two attribute values.

Decomposition. If $A_v \rightarrow B_vC_v$, $A_v \rightarrow C_v$, then $A_v \rightarrow B_v$. This axiom states that an attribute value determines the concatenation of two other attribute values, and then it also determines each of them separately.

Transitivity Rule. If $A_v \rightarrow B_v$, $B_v \rightarrow C_v$, then $A_v \rightarrow C_v$. This axiom states the transitivity of attribute-value dependencies for a categorical attribute.

3.4 Search Space Pruning

Our support pruning is adopted from frequent itemset mining which finds sets of items whose frequency exceeds a given threshold. In our application, the concept of support is measured by the frequency of attribute combinations:

$$Support(X) = \frac{|Dom(X)|}{n} \qquad (1)$$

where n is the cardinality of the relational instance r. It is obvious that if the support of X is lower than S, then the support for any superset of X must also be smaller than S. We can prune X and all of its attribute supersets.

Support Pruning rule: The support of a conditional constraint of multiple attribute value combinations has no more than the support of a single attribute value constraint.

$$Support(X_1, \ldots, X_n) = \min(Support(X_1), \ldots Support(X_n) \qquad (2)$$

3.5 Trusted Machine Learning

In this section, we show theoretically why complex integrity constraints are effective in identifying those tuples for which the learning model may make incorrect predictions. To this end, we define unsafe tuples and show that the "ideal" complex integrity constraint provides a robust and complete mechanism for detecting unsafe tuples.

We formally define the unsafe tuple. We use $[r; Y]$ to denote the labeled dataset obtained by attaching the label attribute Y to the instance r collected by the intelligent railroad system and $Dom(Y)$ to denote the domain of Y.

Unsafe tuple. Given a class C of functions, and a labeled instance $[r; Y] \subset [Dom(r) \times Dom(Y)]$ a tuple $t \in Dom(r)$ is an unsafe tuple w.r.t. C and $[r; Y]$ if $\exists\, f, g \in C\ s.t.\ f(r) = g(r) = Y$ but $f(t) \neq g(t)$.

Intuitively, if there exist two different prediction functions f and g that agree on all tuples in r but not on t, then t is unsafe. Therefore, we can use the following approach for trusted machine learning:

Learn complex integrity constraints Φ for the instance r.

Declare t as unsafe if t does not satisfy Φ.

The above approach is sound and complete for characterizing unsafe tuples. The characterization of unsafe tuples is attributed to the following proposition:

Proposition. There exists a complex integrity constraint Φ for r s.t. The following statement is true: "$\exists \Phi(\mathrm{t})\ iff\ t$ is unsafe w.r.t C and $[r; Y]$ for all $t \in Dom(r)$".

The required complex integrity constraint Φ is $\exists f, g \in C: f(r) = g(r) = Y \implies f(\mathrm{t}) = g(t)$. Intuitively, when all possible pairs of functions that agree on r also agree on t, only then can the prediction on t can be trusted.

4 Experimental setup

We now present experimental evaluations to demonstrate the effectiveness of complex integrity constraints in our case study application: trusted machine learning in intelligent railroad systems (R. Ak et al., 2016). Our experiments address the following research question.

- How effective are complex integrity constraints for plausible machine learning?

- Is there a relationship between the score of constraint violations and the prediction accuracy of ML models?

Implementations: The current prototype of AUDITOR is implemented in Python and with an AMD Ryzen Processor with 3.40 GHZ and 16 GB memory.

Datasets. We use one real subway operation dataset, which is described as follows.

Xiamen Subway. The Xiamen metro data contain 13 attributes : date, time, total miles run, traction energy consumption, regenerative energy consumption, auxiliary energy consumption, brake resistor energy consumption, network flow, network pressure, traction brake value, on-board rate, train speed, station number, and vehicle status. We use a subset of the dataset that contains all information about the vehicle operations in 2018. In this dataset, most of the attributes are numerical attributes (e.g., total miles run, traction energy consumption, on-board rate, etc.)

Evaluation Metrics. We evaluate our complex integrity constraint discovery approach on both PR-AUC (Kieu et al., 2019) and mean absolute error (MAE) (Ranjan et al., 2021). The definition of MAE is described as follows:

$$MAE = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n} \quad (3)$$

where n is the number of sample tuples. In addition, the threshold S used to search for spatial pruning in our experiments took some time. We set the threshold by checking object pairs in the 100 probability subranges. Eventually we obtain the S to 0.3.

In order to quantify how credible the dataset is in general, we designed the Average Violation. The definition of Average Violation is described as follows:

$$Average\ Violation = \frac{|\sum_{i=1}^{n}\{t_i \in r | \neg \Phi(t_i)\}|}{n} \quad (5)$$

where $|\sum_{i=1}^{n}\{t_i \in r | \neg \Phi(t_i)\}|$ denotes the number of unsafe tuples in relation instance r.

Intuitively, the larger the number of unsafe tuples, the more likely the results of model inference will be unreliable. Therefore, the larger the Average Violation value is for a dataset, the higher the ratio of unsafe tuples to the dataset, and the more unreliable the inference result will be using the model to reason about this dataset.

4.1 Applicability

To simulate the actual operation scenario of modern intelligent railroad systems, we divide the dataset into two parts: the training set and the inferred dataset. The training set is used to train the system's deployed model and discover complex integrity constraints. The inferred dataset is used to verify the model performance and the association between unsafe tuples that violate complex integrity constraints.

First, we choose the case of train operating speed prediction with the motivation of applying complex integrity constraints to test the credibility of models deployed on modern intelligent railroad systems. The use of models for predicting train operating speed is a beneficial application, as it assists the driver in making decisions to control the train's current speed (avoiding late trains). The accuracy of the model is critical to the driver's decision-making. Therefore, we need to check in real-time whether the inference results of the model are plausible. With this in mind, we apply complex integrity constraints to analyze the impact of the proportion of unsafe data found during the train's operation on the model's accuracy. Ideally, the complex integrity constraint detects unsafe data when the train collects operational data in real-time before inputting them to the model. Finally, the driver ignores the model's results based on unsafe data inference, thus achieving the safety and effectiveness of modern intelligent transportation systems.

References

Chen, Hongtian, et al., 2020. "Data-driven fault diagnosis for traction systems in high-speed trains: A survey, challenges, and perspectives." IEEE Transactions on Intelligent Transportation Systems. doi: 10.1109/TITS.2020.3029946.

Hu, Q.X., Long, J.S., Wang, S.K., He, J.J., Bai, L., Du, H.L. and Huang, Q.X., 2021. A novel time-span input neural network for accurate municipal solid waste incineration boiler steam temperature prediction. Journal of Zhejiang University-SCIENCE A, 22(10), pp.777-791.

Zhou, P., Li, T., Zhao, C.F. and Zhang, J.Y., 2020. Numerical study on the flow field characteristics of the new high-speed maglev train in open air. Journal of Zhejiang University-SCIENCE A, 21(5), pp.366-381.

Ho, L.V., Nguyen, D.H., de Roeck, G., Bui-Tien, T. and Wahab, M.A., 2021. Damage detection in steel plates using feed-forward neural network coupled with hybrid particle swarm optimization and gravitational search algorithm. Journal of Zhejiang University-SCIENCE A, 22(6), pp.467-480.

L. Zhu, F. R. Yu, Y. Wang, B. Ning and T. Tang, 2019. "Big Data Analytics in Intelligent Transportation Systems: A Survey," in IEEE Transactions on Intelligent Transportation Systems, vol. 20, no. 1, pp. 383-398, Jan., doi: 10.1109/TITS.2018.2815678.

Vikrant Sharma, S.S. Chandel, 2013. Performance and degradation analysis for long term reliability of solar photovoltaic systems: A review, Renewable and Sustainable Energy Reviews, Volume 27, Pages 753-767, https://doi.org/10.1016/j.rser.2013.07.046.

Fariha, A., Tiwari, A., Radhakrishna, A., Gulwani, S. and Meliou, A., 2021, June. Conformance constraint discovery: Measuring trust in data-driven systems. In Proceedings of the 2021 International Conference on Management of Data (pp. 499-512).

Wentao Hu, Dongxiang Zhang, Dawei Jiang, Sai Wu, Ke Chen, Kian-Lee Tan, and Gang Chen. 2020. AUDITOR: A System Designed for Automatic Discovery of Complex Integrity Constraints in Relational Databases. In Proceedings of the International Conference on Management of Data (SIGMOD '20). Association for Computing Machinery, New York, NY, USA, 2697–2700. DOI:https://doi.org/10.1145/3318464.3384683

Bai, Q., Bedi, A.S., Agarwal, M., Koppel, A. and Aggarwal, V., 2022, June. Achieving zero constraint violation for constrained reinforcement learning via primal-dual approach. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 36, No. 4, pp. 3682-3689).

Livshits, E., Kimelfeld, B. and Roy, S., 2020. Computing optimal repairs for functional dependencies. ACM Transactions on Database Systems (TODS), 45(1), pp.1-46.

Wu, P., Yang, W., Wang, H. and Huang, L., 2020, September. GDS: General Distributed Strategy for Functional Dependency Discovery Algorithms. In International Conference on Database Systems for Advanced Applications (pp. 270-278). Springer, Cham.

Fan, W., Hu, C., Liu, X. and Lu, P., 2020. Discovering graph functional dependencies. ACM Transactions on Database Systems (TODS), 45(3), pp.1-42.

Y. Huhtala, J. Kärkkäinen, P. Porkka and H. Toivonen,1999 "Tane: An Efficient Algorithm for Discovering Functional and Approximate Dependencies," in The Computer Journal, vol. 42, no. 2, pp. 100-111, Jan. doi: 10.1093/comjnl/42.2.100.

W. Fan, F. Geerts, J. Li and M. Xiong, 2010. "Discovering Conditional Functional Dependencies," in IEEE Transactions on Knowledge and Data Engineering, vol. 23, no. 5, pp. 683-698, May 2011, doi: 10.1109/TKDE.154.

Sebastian Kruse and Felix Naumann. 2018. Efficient discovery of approximate dependencies. Proc. VLDB Endow. 11, 7, 759–772. DOI:https://doi.org/10.14778/3192965.3192968

L. Caruccio, V. Deufemia and G. Polese, 2016. "Relaxed Functional Dependencies—A Survey of Approaches," in IEEE Transactions on Knowledge and Data Engineering, vol. 28, no. 1, pp. 147-165, doi: 10.1109/TKDE.2015.2472010.

Tobias Bleifuß, Sebastian Kruse, and Felix Naumann. 2017. Efficient denial constraint discovery with hydra. Proc. VLDB Endow. 11, 3 (November 2017), 311–323. DOI:https://doi.org/10.14778/3157794.3157800

Pena, E.H., de Almeida, E.C. and Naumann, F., 2021. Fast detection of denial constraint violations. Proceedings of the VLDB Endowment, 15(4), pp.859-871.

Breve, B., Caruccio, L., Deufemia, V. and Polese, G., 2022. RENUVER: A Missing Value Imputation Algorithm based on Relaxed Functional Dependencies. In EDBT (pp. 1-52).

Berti-Equille, L., Harmouch, H., Naumann, F., Novelli, N. and Saravanan, T., 2018, August. Discovery of genuine functional dependencies from relational data with missing values. In The 44th International Conference on Very Large Data Bases (VLDB) (Vol. 11, No. 8), doi: 10.14778/3204028.3204032

Wei, Z., Leck, U. and Link, S., 2019. Discovery and ranking of embedded uniqueness constraints. Proceedings of the VLDB Endowment, 12(13), pp.2339-2352.

Wei, Z. and Link, S., 2019. Embedded functional dependencies and data-completeness tailored database design. Proceedings of the VLDB Endowment, 12(11), pp.1458-1470.

Wei, Z., Hartmann, S. and Link, S., 2020, June. Discovery algorithms for embedded functional dependencies. In Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data (pp. 833-843).

Kossmann, J., Papenbrock, T. and Naumann, F., 2022. Data dependencies for query optimization: a survey. The VLDB Journal, 31(1), pp.1-22.

Qahtan, A., Tang, N., Ouzzani, M., Cao, Y. and Stonebraker, M., 2020. Pattern functional dependencies for data cleaning. Proc. {VLDB} Endow. 13, 5, 684-697, doi: 10.14778/3377369.3377377

Qahtan, A., Tang, N., Ouzzani, M., Cao, Y. and Stonebraker, M., 2020. Pattern functional dependencies for data cleaning.

Pena, E.H., de Almeida, E.C. and Naumann, F., 2019. Discovery of approximate (and exact) denial constraints. Proceedings of the VLDB Endowment, 13(3), pp.266-278.

Caruccio, L., Deufemia, V., Naumann, F. and Polese, G., 2020. Discovering relaxed functional dependencies based on multi-attribute dominance. IEEE Transactions on Knowledge and Data Engineering, 33(9), pp.3212-3228.

Caruccio, L., Deufemia, V. and Polese, G., 2020. Mining relaxed functional dependencies from data. Data Mining and Knowledge Discovery, 34(2), pp.443-477.

Caruccio, L. and Cirillo, S., 2020. Incremental discovery of imprecise functional dependencies. Journal of Data and Information Quality (JDIQ), 12(4), pp.1-25.

Xu Chu, Ihab F. Ilyas, and Paolo Papotti. 2013. Discovering denial constraints. Proc. VLDB Endow. 6, 13 (August 2013), 1498–1509. DOI:https://doi.org/10.14778/2536258.2536262

Ghosh, S., Lincoln, P., Tiwari, A., Zhu, X. and Edu, W., 2016, June. Trusted machine learning for probabilistic models. In ICML Workshop on Reliable Machine Learning in the Wild.

Toreini, E., Aitken, M., Coopamootoo, K., Elliott, K., Zelaya, C.G. and Van Moorsel, A., 2020, January. The relationship between trust in AI and trustworthy machine learning technologies. In Proceedings of the 2020 conference on fairness, accountability, and transparency (pp. 272-283).

Qolomany, B., Mohammed, I., Al-Fuqaha, A., Guizani, M. and Qadir, J., 2020. Trust-based cloud machine learning model selection for industrial IoT and smart city services. IEEE Internet of Things Journal, 8(4), pp.2943-2958.

W. Chen, F. Guo and F. -Y. Wang, 2015."A Survey of Traffic Data Visualization," in IEEE Transactions on Intelligent Transportation Systems, vol. 16, no. 6, pp. 2970-2984, Dec. doi: 10.1109/TITS.2015.2436897.

P. Bohannon, W. Fan, F. Geerts, X. Jia and A. Kementsietsidis, 2007."Conditional Functional Dependencies for Data Cleaning," IEEE 23rd International Conference on Data Engineering, pp. 746-755, doi: 10.1109/ICDE.2007.367920.

R. Ak, O. Fink and E. Zio, 2016. "Two Machine Learning Approaches for Short-Term Wind Speed Time-Series Prediction," in IEEE Transactions on Neural Networks and Learning Systems, vol. 27, no. 8, pp. 1734-1747, Aug. doi: 10.1109/TNNLS.2015.2418739.

Kieu, T., Yang, B., Guo, C. and Jensen, C.S., 2019, August. Outlier Detection for Time Series with Recurrent Autoencoder Ensembles. In IJCAI (pp. 2725-2732).

N. Malini and M. Pushpa, "Analysis on credit card fraud identification techniques based on KNN and outlier detection," 2017 Third International Conference on Advances in Electrical, Electronics, Information, Communication and Bio-Informatics (AEEICB), 2017, pp. 255-258, doi: 10.1109/AEEICB.2017.7972424.

Azzedine Boukerche, Lining Zheng, and Omar Alfandi. 2020. Outlier Detection: Methods, Models, and Classification. ACM Comput. Surv. 53, 3, Article 55 (May 2021), 37 pages. DOI:https://doi.org/10.1145/3381028

Ranjan, K.G., Tripathy, D.S., Prusty, B.R. and Jena, D., 2021. An improved sliding window prediction ‐ based outlier detection and correction for volatile time ‐ series. International Journal of Numerical Modelling: Electronic Networks, Devices and Fields, 34(1), p.e2816.