GraDit: graph-based data repair algorithm for multiple data edits rule violations

Wa Ode Zuhayeni Madjida and I Gusti Bagus Baskara Nugraha
School of Electrical Engineering and Informatics, Institut Teknologi Bandung, Bandung, Indonesia
E-mail: zuhayeni@students.itb.ac.id

Abstract. Constraint-based data cleaning captures data violation to a set of rule called data quality rules. The rules consist of integrity constraint and data edits. Structurally, they are similar, where the rule contain left hand side and right hand side. Previous research proposed a data repair algorithm for integrity constraint violation. The algorithm uses undirected hypergraph as rule violation representation. Nevertheless, this algorithm can not be applied for data edits because of different rule characteristics. This study proposed GraDit, a repair algorithm for data edits rule. First, we use bipartite-directed hypergraph as model representation of overall defined rules. These representation is used for getting interaction between violation rules and clean rules. On the other hand, we proposed undirected graph as violation representation. Our experimental study showed that algorithm with undirected graph as violation representation model gave better data quality than algorithm with undirected hypergraph as representation model.

1. Introduction
Data is important for an organization to make a decision, so that the organization should keep their data quality. The better the data quality, the data can be used as a basis for decision making. In fact, existing data is always dirty or contain some glitches because of data collecting and recording process. This leads to inconsistent data and could make organization falls on erroneous decisions. One way to eliminate these inconsistencies and ensure the data quality is by doing data repair or data cleaning [1].

Data cleaning is a process to detect and correct errors in data. Data can be relational database tables and records on file. In data cleaning process, we want to make sure that each record in data is clean or not contain faulty value, so that we can use the data for analysis and describe valid conditions. Constraint-based is one of the data cleaning approach.

In constraint-based data cleaning, data is said to be dirty if there are inconsistencies or violations against data quality rules [2]. The rules describe relationship between attributes in data. There are two kind of data quality rules, they are integrity constraint and data edits [8].

Integrity constraint is used to express the data quality rules for checking inconsistencies in relational database. Several types of integrity constraints are functional dependency (FD) and inclusion functional dependency [2], conditional functional dependency (CFD) [3] and conditional inclusion dependency [2] [4], denial constraint [5], matching dependencies [6], and editing-constraints [12].
Furthermore, data edits is used for data where the structure is corresponding to questionnaires schema. National statistics institution uses data edits to keep their data quality by checking inconsistencies in data that come from census questionnaires. Every set of answer in questionnaires must satisfy defined data edits rules, so it can be referred as the correct answer. If there are inconsistent answer, the data must be repaired.

Various algorithm has been proposed for repairing data inconsistencies in constraint-based data cleaning. [9] and [10] introduce a repairing algorithm that gives some cleaning strategies based on cost. [11] and [12] utilize master data for identifying wrong value in an attribute. [13] uses statistical methods for improving accuracy of inconsistent data. [14] takes advantage of user engagement in providing repair updates and using machine learning to identify and apply the correct updates directly to database without user engagement anymore. Another study was conducted by [15] in proposing a data cleaning algorithm for matching and data repair. However, these researches are focus on how repair inconsistencies based on violation of a rule without consider another defined rules.

In [16] and [17], they proposed an algorithm that deal with interaction among defined rules. They represent rules violation into undirected hypergraph to get conflict attribute in data that must be repaired. Nevertheless, the algorithm is designed for integrity constraint FD and CFD, not for data edits rules. Although they have similar structure, the algorithm can not be applied for data edits rules.

Structurally, data edits rules is similar with constant conditional functional dependency (constant-CFD), which incorporate bindings of related values on attribute. Constant value in data edits rules is identical with pattern tableau in CFD. Other related structure is element of the rule that contain left hand side (LHS) and right hand side (RHS). However, we can not represent data edits rule using undirected hypergraph and only for rule violation, due to differences of the rule characteristics.

If there are a set of data edits rules:
\[
\varphi_1 : [A = 1] \land [B = 2] \rightarrow [C \neq \text{null}]
\]
\[
\varphi_2 : [C = 1] \rightarrow [D = 2]
\]
\[
\varphi_3 : [C = 2] \rightarrow [E = \text{null}]
\]
\[
\varphi_4 : [D = 2] \lor [E = 2] \rightarrow [F \neq \text{null}]
\]
\[
\varphi_5 : [F = \text{null}] \land [G = 1] \lor [G = 2] \rightarrow [H = \text{null}]
\]

**Problem 1.** LHS of data edits rules contains one or more operands, AND (\(\land\)) and OR (\(\lor\)). In FD or CFD, operand in LHS is only AND. For example, data edits rules \(\varphi_5\) contains two operands. Undirected hypergraph can not handle these operands, so that conflict attribute in repairing process falls into wrong attribute.

**Problem 2.** A set of data edits rules contain rules that organize various attribute values based on attribute domain. Consider attribute \(C\) has domain value \(\text{dom}(C) = \{1, 2\}\), \(\varphi_2\) and \(\varphi_3\) are rules which determine \(D\) value based on all possibility of \(C\) value. If we just represent rule violations, we can not get repair expression of cleaning process because it is at any other rule.

**Problem 3.** One rule and another rule in data edits rules are linked each other. RHS in one rule can be LHS in another rule, i.e RHS [\(F \neq \text{null}\) ] in \(\varphi_4\) is LHS in \(\varphi_5\). By using undirected hypergraph and representing only for rule violation, we can not get interaction between violation rule and clean rule, so that repair one attribute can create other new violation.

These problems motivate the study of novel repair algorithm for data edits rule. We proposed another kind of violation representation and designing the repair algorithm.

2. Related Work
Constraint-based data cleaning is one of data cleaning approach. Several algorithm are proposed for repairing data inconsistencies in constraint-based data cleaning. [9] proposed a data cleaning
method, sampling against space of possible repairs. [10] uses cost in comparing data and constraint repair. [11] and [12] utilize master data for identifying wrong value in an attribute. [13] improves accuracy of inconsistent data by using statistical methods. [14] takes advantage of user engagement in providing repair updates. The update is stored in database and will be used by machine learning algorithm to identify and apply the correct updates without user engagement anymore. Another study was conducted by [15] proposed data cleaning algorithm for matching and data repair. However, these research focus on repairing one violation without considering interaction among rules. This can make the repair is not based on other attribute value in different rule, so that it causes new violation of other rule.

In [16], they proposed an algorithm that consider the interaction between violation rules. The violation is represented in undirected hypergraph and using minimum vertex cover (MVC) to define a conflict attribute, namely attribute that must be repaired. The proposed algorithm repair one attribute in every iteration until there is no violation in data. To do the repair holistically, [17] continued this study. The research takes advantage of the undirected hypergraph for getting context of conflict attribute. The context are the attribute involvement in other violation rule and its change effect for the other attribute in same violation rule. They used this context to build repair expression and doing repair for several attributes in every iteration (holistic repair). To determine the value of attribute improvements, researchers used Value Frequency Map (VFM) and Quadratic Programming (QP). However these algorithms are designed for integrity constraint FD and CFD, and can not be applied for data edits rules, so that we proposed a repair algorithm for data edits rules.

3. Proposed Algorithm

3.1. Data Edits Rule

National statistics using data edits for checking inconsistencies in data that come form census questionnaires. An example of using data edits for checking data inconsistencies is as follows.

An inconsistent answer in a questionnaire can be declared as:

marital status = "married", age = "5 years old"

To detect this violations, rule can be made as:

if marital status is married, age must not be less than 16 years old

Edits express the inconsistency condition, namely:

marital status = "married" \( \land \) age < 16

Based on the rule, we can define a pattern that must be satisfied by data, as follow:

if marital status is married, then age must greater or equal than 16 years old

Married is constant for marital status, and greater than 16 years old is constant for age. Denoted that \( \varphi \) is data edits rule, which refer to rules that must be satisfied by data, then the rule mathematically can be noted as:

\[
\varphi : [\text{marital} = "\text{married}"] \rightarrow [\text{age} \geq 16].
\]
Based on the rule above, the LHS of the rule is \([\text{marital} = \text{"married"}]\) and the RHS is \([\text{age} \geq 16]\). LHS and RHS of the rule can contain more than one expression and implicate AND \((\land)\) and OR \((\lor)\) operand.

A record in data is said violating the rule, if the record meets the condition at LHS, but does not meet the condition on the RHS.

3.2. Proposed Algorithm

In data that come from questionnaire schema, every record must satisfy defined data edits rules. Consequently, violation detection and data repairing are processed per record. Figure 1 shows proposed data cleaning algorithm based on data edits rules. We called it GraDit. Here is step by step of the algorithm:

**Step 1**: Parsing Rules. This process would generalize all defined rules. Expression in a rule that hold != (not equal), <, ≤, >, ≥ is elaborated into equality expression based on domain attribute value. Additionally, if a rule contains two operands AND and OR, the rule is converted to disjunctive normal form (DNF) by using distributive law.

**Step 2**: Generate Bipartite-Directed Hypergraph. This process represents overall defined rules and attributes in bipartite-directed hypergraph. Node of graph consist of two set, that is attribute nodes and rule nodes. Attribute nodes represent attribute in a rule and rule nodes represent rule it self. This representation would be used for getting interaction between violation rule and clean rule in repairing process.

For every record in data, do step 3 until step 7. This process will continue until number of current violation is greater than previous violation.

**Step 3**: Hit data. Set flag in bipartite-directed hypergraph based on value of each attribute in data.

**Step 4**: Identify violation. This process would detect rule violation on data by using edits.

**Step 5**: Generate conflict graph. Generating conflict graph, namely graph that represent the violations. We use undirected graph as violation representation model. Node in the graph is attribute on violation rule, and edge connects each node based on relationship characteristics between attribute in the same rule.

**Step 6**: Identify MVC. Identify MVC of the conflict graph by using vertex cover approximation algorithm. Node that is selected as MVC would become repaired attribute candidate in repairing process.

**Step 7**: Repair data. We proposed algorithm 1 for repairing data. This process would get attribute nodes in bipartite-directed hypergraph, namely attributes that were selected as repair candidate. The process would find attribute nodes which have value besides value attribute of data. These nodes is evaluated based on some parameters and the best nodes would become repair expression. This study proposed some parameters in selecting repair expression, they are:

(i) Number of ingoing degree that come from rule node that has error flag in bipartite-directed hypergraph (IER).

(ii) Number of ingoing degree that come from flagged attribute node to connected error rule node (IHV).

(iii) Number of outgoing degree to rule node that has outgoing edge to flagged attribute node (OHR).

(iv) Number of outgoing degree to flagged attribute node from connected rule node (OHV).

Table 1 shows algorithm comparison between previous research and this study.
Algorithm 1 Data Repairing

**Input:** vertex list from mvc (conf), data, bipartite-directed hypergraph (graph)

**Output:** Repair data

1: repairCandidate ← []
2: IER ← []
3: IHV ← []
4: OHR ← []
5: OHV ← []
6: for all \( vertex \in \text{conf} \) do
7: candidate ← get variable nodes of \( vertex \) in \( graph \) besides hitdata
8:  repairCandidate ← repairCandidate ∪ candidate
9:  for all \( rc \in \text{repairCandidate} \) do
10:     er ← compute number of inbound error rule node to \( rc \)
11:     IER.add(er)
12:     hv ← compute number of inbound hit variable node to error rule node of \( rc \)
13:     IHV.add(hv)
14:     hr ← compute number of outbound rule node of \( rc \) which has outbound hit variable node
15:     OHR.add(hr)
16:     hv ← compute number of outbound hit variable node from outbound rule node of \( rc \)
17:     OHV.add(hv)
18:  end for
19: end for
20: repair ← get repairCandidate with the highest number in IER
21: if repair.size() = 1 then
22:    data.update(repair) \( \triangleright \) repair data with repairCandidate with the highest number in IER
23: else
24:    repair ← get repair with the highest number in IHV
25:    if repair.size() = 1 then
26:       data.update(repair)
27:    else
28:       repair ← get repair with the highest number in OHR
29:       if repair.size() = 1 then
30:          data.update(repair)
31:       else
32:          repair ← get repair with the highest number in OHV
33:          data.update(repair)
34:      end if
35:    end if
36: end if
37: return data
4. Experiments

4.1. Experimental Settings

Datasets and rules. We use national socio-economic survey datasets which is collected by Statistics of Indonesia. Datasets consist of 100 record and 209 attributes where 149 attributes of them are set in rules. The datasets is divided into three instances:

(i) Clean Data $I_c$. This data is cleaned data that has satisfy all defined rules. We use this data as ground truth.

![Flowchart of GraDit Algorithm](image)  

**Figure 1.** GraDit Algorithm.
Table 1. Algorithm Comparison of Previous Research and Proposed.

| Algorithm                  | FindVRepair [16] | Holistic Repair [17] | GraDit |
|----------------------------|------------------|----------------------|--------|
| Data quality rule          | FD and CFD       | FD and CFD           | Data edits rule |
| Representation for overall rules | NO               | NO                  | YES    |
| Violation representation model | undirected hypergraph | undirected hypergraph | undirected graph |
| Repair value of conflict attribute | using other attribute value in same rule | using Value Frequency Map (VFM) and Quadratic Programming (QP) | using proposed parameters |

(ii) Dirty Data $I_d$, data that has some rule violations. This data comes from $I_c$ where some attribute value are changed by a certain error rate, that is, ratio of the number of dirty attribute to the total number of attribute which is set in the rule. Error rate which would be generated is 10 to 40 percent. Error is generated randomly on the attributes of the datasets based on domain value of each attribute.

(iii) Repaired Data $I_r$. This data is repair of $I_c$, that is output of repair algorithm.

There are 483 data edits rule that manage relationship between attribute of dataset. These rules consist of 286 rules that manage relationship between two attributes (no operand in rule), 131 rules with one operand AND or OR in a rule, and 66 rules that consist of two operand AND and OR in a rule. One rule and other rules are linked each other.

**Algorithms.** We implement two algorithms, they were proposed algorithm in figure 1 and proposed algorithm where the violation representation is changed into undirected hypergraph. Both of them have been implemented in Java. Based on these algorithm, we would show how violation representation takes effect to data repairing result.

**Metric.** We use F-measure to measure repair quality. The formula of F-measure is $(2x(PxR)/(P + R))$, where P is precision and R is recall. This metric is adopted from [9].

4.2. Experimental Results

![Figure 2. F-Measure Maximal Value of Data Repairing Based on Error Rate.](image1)

![Figure 3. Percentage of Record that Have F-Measure >0.5 Based on Error Rate.](image2)
4.2.1. **F-Measure Maximal Value of Data Repairing Based on Error Rate**

Figure 2 shows maximal value of f-measure that is obtained from implemented algorithm. Out of 100 tested record on the error rate of 10%, f-measure can reach maximum value 1 for the algorithm by using undirected graph as a violation representation model. In other words, the algorithm can fix all the errors properly. This is because the generated error is an attribute on RHS of rule, so that violations can be captured and repaired overall by the algorithm. However, although the algorithm can provide maximum value, algorithm by using undirected hypergraph as violation representation can not deliver maximum value 1.

Violations that are represented using undirected hypergraph lead to mistake in identifying attribute candidates to be repaired. In violation rule, there is an LHS attribute that exist in almost every rule that were violated. This makes the attribute become vertex cover when calculating MVC. In fact, the attribute is not error attribute.

In data with error rate 20 to 40%, f-measure of algorithm with undirected graph violation representation is close to 1. In contrast, the algorithm by using undirected hypergraph, f-measure decline close to 0 by increasing the error rate in the data.

F-measure value of each repaired record is greatly affected by violation representation model. This representation model determine attribute candidates to be repaired. Inaccuracy in defining repaired candidate attribute, it would affect repaired data quality. Undirected graph can give better results than undirected hypergraph because undirected graph representation consider relationship characteristic of attributes in a rule, which is based on the operand in the rule, either AND nor OR. Otherwise, undirected hypergraph can not describe attribute relationship characteristic because all attributes in rule that become node on the hypergraph were covered by one edge.

Other factor that influences the value of f-measure is generated error attribute. The attribute can be involved in LHS of a rule, RHS, or LHS in one rule and RHS in another rule. If the error attribute is only in LHS and error value of the attribute is not a LHS conditions that must be met by the rule, then the error can not be captured by the violations. The other case, there were conditions that are satisfied by LHS attributes, then led the RHS attributes captured as a violation. The RHS attributes have possibility to be repaired by algorithm. These improvements is not a proper repair based on ground truth, but in terms of cost, which is the number of attribute change in data repairing process, it is an improvement with the lowest cost.

4.2.2. **Percentage of Record that Have F-Measure >0.5 Based on Error Rate**

Based on percentage of record that have f-measure value above 0.5, namely data that can be corrected more than 50% of the overall error attribute, undirected graph representation could give significant result than algorithm by using undirected hypergraph. Figure 3 shows the results. Algorithms with undirected hypergraph does not produce any of record that have f-measure above 0.5, either on the data with an error rate of 10% to 40%. In contrast, undirected graph algorithms can provide 80% of record that have f-measure above 0.5 in data with 10% of error rate. This percentage continues to decrease with higher error rate in the data.

5. **Conclusion**

In this paper, we proposed an algorithm for constraint-based data cleaning of data edits rule. We introduce bipartite-directed hypergraph as model representation of overall data edits rule for getting interaction between violation rule and clean rule. For representing rules violation, we use undirected graph that could describe attribute relationship characteristic in a rule. Based on experintal study, algorithm that using undirected graph as violation representation could give higher f-measure than algorithm which undirected hypergraph. Out of 100 tested record by using undirected graph algorithm, 80% records give f-measure above 0.5 in data with 10% error rate, 63% records in data with 20% error rate, 41% records in data with 30% error rate, and
31% records in data with 40% error rate. This result shows that by representing violation in undirected graph, the proposed algorithm can give better repair quality than using undirected hypergraph.

References
[1] T. Dasu and T. Johnson, Exploratory Data Mining and Data Cleaning. New York, NY, USA: John Wiley & Sons, Inc., 1 ed., 2003.
[2] W. Fan and F. Geerts, Foundations of Data Quality Management. Morgan & Claypool Publishers, 2012.
[3] W. Fan, F. Geerts, X. Jia, and A. Kementsietsidis, “Conditional functional dependencies for capturing data inconsistencies,” ACM Trans. Database Syst., vol. 33, pp. 6:1–6:48, June 2008.
[4] P. Z. Yeh and C. A. Puri, “Discovering conditional functional dependencies to detect data inconsistencies,” in Proceedings of the Fifth International Workshop on Quality in Databases at VLDB2010, vol. 176, Citeseer, 2010.
[5] X. Chu, I. F. Ilyas, and P. Papotti, “Discovering denial constraints,” Proc. VLDB Endow., vol. 6, pp. 1498–1509, Aug. 2013.
[6] W. Fan, X. Jia, J. Li, and S. Ma, “Reasoning about record matching rules,” Proc. VLDB Endow., vol. 2, pp. 407–418, Aug. 2009.
[7] W. Fan, J. Li, S. Ma, N. Tang, and W. Yu, “Towards certain fixes with editing rules and master data,” The VLDB Journal, vol. 21, no. 2, pp. 123–238, 2012.
[8] M. Carey, S. Ceri, P. Bernstein, U. Dayal, C. Faloutsos, J. Freytag, G. Gardarin, W. Jonker, V. Krishnamurthy, M. Neimat, et al., Data-Centric Systems and Applications. Springer, 2006.
[9] G. Besjales, I. F. Ilyas, and L. Golab, “Sampling the repairs of functional dependency violations under hard constraints,” Proc. VLDB Endow., vol. 3, pp. 197–207, Sept. 2010.
[10] F. Chiang and R. J. Miller, “A unified model for data and constraint repair,” in 2011 IEEE 27th International Conference on Data Engineering, pp. 446–457, April 2011.
[11] T. Diallo, J.-M. Petit, and S. Servigne, “Discovering editing rules for data cleaning,” in Proceedings of AQB conference, p. 40, 2012.
[12] W. Fan, J. Li, S. Ma, N. Tang, and W. Yu, “Towards certain fixes with editing rules and master data,” The VLDB Journal, vol. 21, no. 2, pp. 213–238, 2012.
[13] G. Cong, W. Fan, F. Geerts, X. Jia, and S. Ma, “Improving data quality: Consistency and accuracy,” in Proceedings of the 33rd International Conference on Very Large Data Bases, VLDB ’07, pp. 315–326, VLDB Endowment, 2007.
[14] M. Yakout, A. K. Elmagarmid, J. Neville, M. Ouzzani, and I. F. Ilyas, “Guided data repair,” Proc. VLDB Endow., vol. 4, pp. 279–289, Feb. 2011.
[15] W. Fan, J. Li, S. Ma, N. Tang, and W. Yu, “Interaction between record matching and data repairing,” in Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data, SIGMOD ’11, (New York, NY, USA), pp. 469–480, ACM, 2011.
[16] S. Kolahi and L. V. S. Lakshmanan, “On approximating optimum repairs for functional dependency violations,” in Proceedings of the 12th International Conference on Database Theory, ICDT ’09, (New York, NY, USA), pp. 53–62, ACM, 2009.
[17] X. Chu, I. F. Ilyas, and P. Papotti, “Holistic data cleaning: Putting violations into context,” in 2013 IEEE 29th International Conference on Data Engineering (ICDE), pp. 458–469, April 2013.
[18] A. S. Asratian, T. M. Denley, and R. Häggkvist, Bipartite graphs and their applications, vol. 131. Cambridge University Press, 1998.
[19] G. Gallo, G. Longo, S. Pallottino, and S. Nguyen, “Directed hypergraphs and applications,” Discrete Appl. Math., vol. 42, pp. 177–201, Apr. 1993.
[20] G. Ausiello, P. G. Franciosa, and D. Frigioni, “Directed hypergraphs: Problems, algorithmic results, and a novel decremental approach,” in Proceedings of the 7th Italian Conference on Theoretical Computer Science, ICTL ’01, (London, UK, UK), pp. 312–327, Springer-Verlag, 2001.
[21] N. J. Nilsson, Problem-Solving Methods in Artificial Intelligence. McGraw-Hill Pub. Co., 1971.
[22] C. zturan, “On finding hypercycles in chemical reaction networks,” Applied Mathematics Letters, vol. 21, no. 9, pp. 881 – 884, 2008.
[23] E. Mendelson, Introduction to Mathematical Logic. CRC Press, 2015.
[24] G. Ausiello, A. D’Atri, and D. Saccà, “Graph algorithms for functional dependency manipulation,” J. ACM, vol. 30, pp. 752–766, Oct. 1983.
[25] P. Cohn, Basic Algebra: Groups, Rings and Fields. Springer London, 2004.