ANMAT: Automatic Knowledge Discovery and Error Detection through Pattern Functional Dependencies

Abdulhakim Qahtan
QCRI, HBKU
aqahtan@hbku.edu.qa

Nan Tang
QCRI, HBKU
ntang@hbku.edu.qa

Mourad Ouzzani
QCRI, HBKU
mouzzani@hbku.edu.qa

Yang Cao
University of Edinburgh
yang.cao@ed.ac.uk

Michael Stonebraker
CSAIL, MIT
stonebraker@csail.mit.edu

ABSTRACT

Knowledge discovery is critical to successful data analytics. We propose a new type of meta-knowledge, namely pattern functional dependencies (PFDs), that combine patterns (or regex-like rules) and *integrity constraints* (ICs) to model the dependencies (or meta-knowledge) between partial values (or patterns) across different attributes in a table. PFDs go beyond the classical functional dependencies and their extensions. For instance, in an employee table, ID "F-9-107", "F" determines the financial department, and "9" determines one's grade. Moreover, a key application of PFDs is to use them to identify erroneous data; tuples that violate some PFDs. In this demonstration, attendees will experience the following features: *PFD discovery* – automatically discover PFDs from (dirty) data in different domains; and *Error detection with PFDs* – we will show errors that are detected by PFDs but cannot be captured by existing approaches.

1 INTRODUCTION

Patterns (or regex-like rules) are widely used to discover meta-knowledge in a given domain, *e.g.,* a Year column should contain *only* four digits. In addition, *integrity constraints* (ICs) have been extensively studied to model data dependencies across columns, *e.g.,* Postal Code uniquely determines City, which can then be used for error detection, query optimization, and data modeling, among others. Our key observation is that by relaxing the limitation of previous ICs, namely the need to operate on entire attribute values, we can specify a new type of data dependencies that can capture partial attribute values that follow some patterns.

	name	gender
r_1:	John Charles	M
r_2:	John Bosco	M
r_3:	Susan Orlean	F
r_4:	Susan Boyle	M
		F

Table 1: D_1: A Name Table

	zip	city
s_1:	90001	Los Angeles
s_2:	90002	Los Angeles
s_3:	90003	Los Angeles
s_4:	90004	New York
		Los Angeles

Table 2: D_2: A Zip Table

Consider two datasets D_1 and D_2, for two tables Name and Zip, respectively. Table Name (Table 1) is defined over the schema (name, gender), and table Zip (Table 2) is defined over the schema (zip, city). Erroneous cells, r_4[gender] in D_1 and s_4[city] in D_2, are highlighted. Their correct values (or ground truth) are F and Los Angeles, which are also shown in the tables, below the erroneous values.

Our Methodology. Our proposed ICs are based on patterns of partial attribute values, as shown below:

λ_1 : **Name** ([name = John\␣\A*] → [gender = M])
λ_2 : **Name** ([name = Susan\␣\A*] → [gender = F])
λ_3 : **Zip** ([zip = 900\D{2}] → [city = Los Angeles])
PFDs

where λ_1/λ_2 says that if someone's first name is John/Susan, then the gender is M/F (\A matches any alphabet and \A* matches any string, which will be defined in Section 2); and λ_3 says that if a five-digit zip code starts by 900, then the city is Los Angeles (\D{2} matches any two consecutive digits). Clearly, λ_2 can detect error r_4[gender] in D_1 and λ_3 can detect error s_4[city] in D_2.

Alternatively, consider two other constraints as follows:

λ_4 : **Name** ([name = $\overline{\backslash\text{LU}\backslash\text{LL}*\backslash\text{␣}}$ \A*] → [gender])
λ_5 : **Zip** ([zip = $\overline{\backslash\text{D}\{3\}}$ \D{2}] → [city])
PFDs

where λ_4 says that one's first name uniquely determines one's gender for table Name (\LU matches any upper case letter and \LL* matches any consecutive lower case letters); and λ_5 states that the first 3 digits of a 5-digit zip code determines

SIGMOD'19, June 30 - July 5, 2019, Amsterdam, The Netherlands
© 2019 Association for Computing Machinery.
ACM ISBN 123-4-5678-9876-7/18/06…$15.00
https://doi.org/10.xxx/123456.123456

the city for table Zip. These two PFDs (λ_4 and λ_5) are defined over two tuples: for example, two tuples match the LHSof λ_4, if they both satisfy the pattern \LU\LL*\␣\A*, and their first names are the same, which is enforced by $\overline{\backslash\mathsf{LU}\backslash\mathsf{LL}{*}\backslash\text{␣}}$.

λ_4 can detect the error r_4[gender] by comparing tuples r_3 and r_4: r_3 and r_4 have the same first name Susan but different gender, which identifies a violation consisting of four cells (r_3[name], r_3[gender], r_4[name], r_4[gender]). Similarly, λ_5 can detect the error s_4[city] by comparing s_4 with either s_1, s_2, or s_3.

The Limitations of the Prior Art. The fundamental limitation of previous ICs (*e.g.,* FDs [**?**] and CFDs [**?**]) is that they enforce data dependencies using the entire attribute values. Consequently, they cannot specify the fine-grained semantics found in partial attribute values.

Our Proposed Demonstration. This demo implements Anmat[1], a system to discover PFDs directly from dirty data, and to use them for error detection as a key application for such ICs. The audience will be able to see PFDs discovered from diverse domains. It will also see how new (*i.e.,* cannot be detected by other ICs) data errors can be detected.

2 PATTERN FUNCTIONAL DEPENDENCIES

Before demonstrating the discovery of PFDs, we need to formally define them. We first discuss the (regex-like) patterns that we use for modeling the partial attribute values. While the class of general regular expressions can be used, it is actually too large for our purpose. In addition, it complicates the problems (*i.e.,* high time complexity) of discovering and applying PFDs, *e.g.,* checking the equivalence of two regular expressions is PSPACE-complete [**?**]. Fortunately, for the purpose of data cleaning, simple patterns are typically sufficient, as it has been shown in recent works [**? ?**].

We use the *generalization tree*, which is a tree defined over an alphabet Σ, where each leaf node is a character in Σ and each intermediate node is a generalization of its child nodes, depicted in Figure 1. It contains upper case letters [A-Z], lower case letters [a-z], digits [0-9], and other symbols. Here, ϵ represent the empty string.

Patterns. A *pattern* P is a sequence of characters defined over the generalization tree. For strings α and β, $\alpha\{N\}$ means N repetitions of α, α & β is the logical **and** of α and β, $\alpha+$ means one-or-more repetitions, and the Kleene star operator $\alpha*$ denotes zero-or-more repetitions. We do not consider *recursive patterns* such as $(\alpha+)*$.

Employing a simple definition of patterns, in contrast to complicated regular expressions, has many benefits as they are: (1) easy to specify, (2) easy to discover, (3) easy to apply, (4) easy to reason about, and (5) most importantly, enough to detect most errors that more general regular expressions can capture in practice.

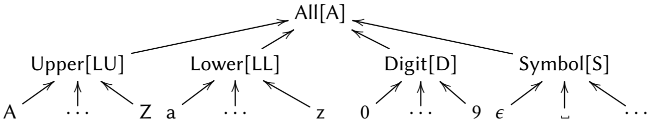


Figure 1: A Generalization Tree

We say that a string s *matches* (or satisfies) a pattern P, denoted by $s \mapsto P$, if s is evaluated to be *true* by P.

Given two patterns P and P', we say that P is *contained* by P', denoted by $P \subseteq P'$, *iff* for any string s, $s \mapsto P$ implies $s \mapsto P'$. In other words, P' is more general than P.

Example 1: [***Patterns.***] Consider zip code 90001 and two patterns $P_1 = \backslash\mathsf{D}\{5\}$ and $P_2 = \backslash\mathsf{D}*$. We have $90001 \mapsto P_1$, $90001 \mapsto P_2$, and $P_1 \subseteq P_2$. □

Constrained Patterns. A *constrained pattern* $\overline{Q}$ is a concatenation of several patterns where at least one is *constrained* (or annotated) by the symbol "$\overline{\ \ }$". We call Q the *embedded pattern* of the constrained pattern $\overline{Q}$. Given a string s, s *matches* a constrained pattern $\overline{Q}$, denoted by $s \mapsto \overline{Q}$, *iff* $s \mapsto Q$.

Given two constrained patterns $\overline{Q}$ and $\overline{Q'}$, we say that $\overline{Q}$ is a *restricted* pattern of $\overline{Q'}$, denoted by $\overline{Q} \subseteq \overline{Q'}$, if for any two strings s, s', $s \equiv_{\overline{Q}} s'$ implies $s \equiv_{\overline{Q'}} s'$.

Example 2: [***Constrained Patterns.***] One example constrained pattern is $\overline{Q_1} = \overline{\backslash\mathsf{LU}\backslash\mathsf{LL}{*}\backslash\text{␣}}\ \backslash\mathsf{A}*$ from the constraint λ_4 presented in the introduction. It is used on the name attribute to enforce the matching over the first name. Another example is $\overline{Q_2} = \overline{\backslash\mathsf{LU}\backslash\mathsf{LL}{*}\backslash\text{␣}}\ \backslash\mathsf{A}{*}\text{␣}\overline{\backslash\mathsf{LU}\backslash\mathsf{LL}{*}}$, which can be used to enforce the matching over both the first name and the last name, but with an arbitrary number of middle names.

The embedded patterns of $\overline{Q_1}$ and $\overline{Q_2}$ are \LU\LL*\␣\A* and \LU\LL*\␣\A*␣\LU\LL*, respectively. Obviously, $Q_2 \subseteq Q_1$, *i.e.,* pattern Q_2 is contained by Q_1, and $\overline{Q_2} \subseteq \overline{Q_1}$, *i.e.,* $\overline{Q_2}$ is a restricted constrained pattern of $\overline{Q_1}$.

Consider two names in Table 1, r_1[name] = John Charles and r_2[name] = John Bosco. We have r_1[name] $\mapsto \overline{Q_1}$, r_2[name] $\mapsto \overline{Q_1}$. Moreover, we have r_1[name] $\equiv_{\overline{Q_1}}$ r_2[name], because r_1[name]($\overline{Q_1}$) = {John}, r_2[name]($\overline{Q_1}$) = {John}, and r_1[name]($\overline{Q_1}$) $\cap$ r_2[name]($\overline{Q_1}$) = {John} $\neq \emptyset$. □

Pattern Functional Dependencies (PFDs). A PFD ψ defined over schema R is a pair $R(X \longrightarrow Y, T_p)$, where:

(1) X and Y are sets of attributes from R,

(2) $X \longrightarrow Y$ is a standard FD, called an *embedded* FD, and

(3) T_p is a tableau with all attributes in X and Y, where for attribute A in X or Y and each tuple $t_p \in T_p$, $t_p[A]$ is either a constrained pattern that matches values in dom(A), or an unnamed variable '$\perp$' that serves as a wildcard.

Please refer to λ_1–λ_5 in Section 1 for PFD examples.

[1] From the Arabic word, patterns.

3 DISCOVERY AND ERROR DETECTION

PFD Discovery. The PFD Discovery algorithm is shown in Figure 2. Given a table and a function to decide whether a set of value pairs forms a PFD as input, it outputs a set of PFDs. The algorithm first profiles the data to prune attributes for which PFDs cannot be found (line 1). For example, we drop all columns with pure numerical values. We then assume that all column pair combinations are potential dependencies for the PFDs. Then for each candidate dependency, the algorithm checks whether there are patterns that can be used to form a PFD (lines 3–14). The same process can be used to work either on tokens (obtained using the function **Tokenize**) or n-grams (using the function **NGrams**) (lines 6,7). Then for each token or n-gram of $t[A]$ (line 6), the algorithm inserts a key-value pair for the token or n-gram into an inverted list, where the key is the token or n-gram of $t[A]$, and the value is a triple consisting of tuple id, position of the token or n-gram in $t[A]$, and $t[B]$ (line 8). Afterwards, it will scan all entries in the inverted list (line 10), and decide which entry can form a meaningful pattern tuple based on a predefined function (lines 11–12).

Error Detection using PFDs. Given a PFD ψ defined over schema R as $(A \rightarrow B, t_p)$, we consider two cases for error detection: constant PFDs, *i.e.,* the constrained parts of the tableau in the B attribute contains only constants, and variable PFDs, *i.e.,* the value related to B attribute in the tableau contains a wildcard. For each constant PFD, we simply do the following scan the table and check, for each tuple t, if $t[A] \mapsto t_p[A]$ and $t[B] \neq t_p[B]$, then there is a violation. In this case, if we assume that the LHS value is correct then the RHS could repaired by changing it to $t_p[B]$. For better performance, we create an index supporting regular expressions for each column present on the LHS of the PFDs. In this case, the search for violations will be limited to those tuples that match $t_p[A]$. For variable PFDs, *i.e.,* $t_p[B] = \perp$, the brute force approach would be to enumerate all possible tuple pairs (t_i, t_j) and check for violations, *i.e.,* $t_i[A] = t_j[A] = t_p[A]$ and $t_i[B] \neq t_j[B]$. Again, we create an index supporting regular expressions for each column present on the LHS of the PFDs to limit the check to only tuples matching $t_p[A]$. However, this is still quadratic. The quadratic time complexity can be avoided using blocking [?].

4 DEMONSTRATION OVERVIEW

Datasets. We will use real-world datasets, from data.gov and ChEMBL (https://www.ebi.ac.uk/chembl/downloads), as well as anonymized private datasets from the MIT data warehouse and local companies in Qatar. The audience is also encouraged to bring its own data and test it using Anmat.

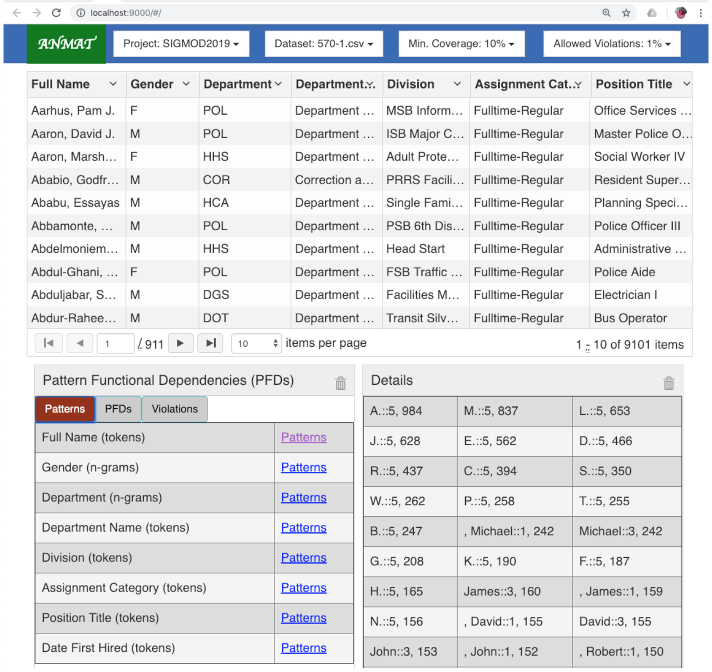
```
Algorithm Discover PFDs
Input:   a relational table $T$,
         a function $f$ and a minimum coverage threshold
         $\gamma$ to make PFD decisions
Output:  a set $\Psi$ of PFDs

1.  $\Phi$ := CandidateDependecies($T$)
2.  $\Psi := \emptyset$            /* the set of discovered PFDs */
3.  for each FD $\varphi : (A \rightarrow B) \in \Phi$ do
4.    $\mathcal{H} := \emptyset$          /* a hash-based inverted list */
5.    for each tuple $t \in T$ do
6.      for each $s \in$ Tokenize($t[A]$)|NGrams($t[A]$) do
7.        for each $u \in$ Tokenize($t[B]$)|NGrams($t[B]$) do
8.          $\mathcal{H}$.insert($s$, (id($t$), $pos_s$, $u$, $pos_u$))
9.    $T_p = \emptyset$ for a new PFD $\psi : (A \rightarrow B, T_p)$
10.   for each entry $h \in \mathcal{H}$ do
11.     if $f(h)$ is true then
12.       add a tuple $t_p$ to $T_p$, w.r.t. entry $h$
13.   if coverage($T_p$) $\geq \gamma$ then
14.     $\Psi := \Psi \cup \{\psi\}$
15. return $\Psi$
```

Figure 2: Algorithm for Discovering PFDs

Parameter Setting. Anmat accepts two user input parameters, namely: (1) *the minimum coverage* and (2) *the ratio of allowed violations*. The minimum coverage represents the ratio of the records that participate in a PFD to the total number of records in the attribute. The participation is determined by checking all the records containing at least one of the patterns that appear in the tuples of the tableau. Since we assume the data is dirty, we tolerate a specific ratio of violations, which are reported as errors. The minimum coverage and the allowed violations give the user the ability to control the number of discovered dependencies. Both parameters represent a trade-off between discovering more dependencies and reducing the rate of false positives. For example, using smaller percentage for the coverage will allow to report more dependencies but it will report more dependencies which are false positives.

System Interface. We have implemented Anmat with two interfaces for different users: a GUI for *lay users* as shown in Figure 3, and a Jupyter Notebook for *programmers*. We will mainly demonstrate the GUI.

Dataset Specification. The user of Anmat will select the project and the dataset to work on from drop-down menus as shown at the top of Figure 3. New users can create their own projects and upload the datasets that need to be processed. After uploading the dataset and setting the minimum coverage and allowed violations, the system will automatically profile the dataset, extract the PFDs, and store the results in a MongoDB database.



localhost:9000/#/

Project: SIGMOD2019 ▾ | Dataset: 570-1.csv ▾ | Min. Coverage: 10% ▾ | Allowed Violations: 1% ▾

Full Name	Gender	Department	Department…	Division	Assignment Cat…	Position Title
Aarhus, Pam J.	F	POL	Department …	MSB Inform…	Fulltime-Regular	Office Services …
Aaron, David J.	M	POL	Department …	ISB Major C…	Fulltime-Regular	Master Police O…
Aaron, Marsh…	F	HHS	Department …	Adult Prote…	Fulltime-Regular	Social Worker IV
Ababio, Godfr…	M	COR	Correction a…	PRRS Facili…	Fulltime-Regular	Resident Super…
Ababu, Essayas	M	HCA	Department …	Single Fami…	Fulltime-Regular	Planning Speci…
Abbamonte, …	M	POL	Department …	PSB 6th Dis…	Fulltime-Regular	Police Officer III
Abdelmoniem…	M	HHS	Department …	Head Start	Fulltime-Regular	Administrative …
Abdul-Ghani, …	F	POL	Department …	FSB Traffic …	Fulltime-Regular	Police Aide
Abduljabar, S…	M	DGS	Department …	Facilities M…	Fulltime-Regular	Electrician I
Abdur-Rahee…	M	DOT	Department …	Transit Silv…	Fulltime-Regular	Bus Operator

1 / 911 10 items per page 1 - 10 of 9101 items

Pattern Functional Dependencies (PFDs)

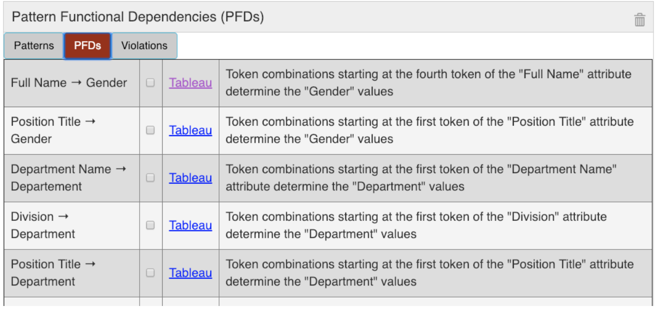
Patterns | PFDs | Violations

Full Name (tokens)	Patterns
Gender (n-grams)	Patterns
Department (n-grams)	Patterns
Department Name (tokens)	Patterns
Division (tokens)	Patterns
Assignment Category (tokens)	Patterns
Position Title (tokens)	Patterns
Date First Hired (tokens)	Patterns

Details

A.::5, 984	M.::5, 837	L.::5, 653
J.::5, 628	E.::5, 562	D.::5, 466
R.::5, 437	C.::5, 394	S.::5, 350
W.::5, 262	P.::5, 258	T.::5, 255
B.::5, 247	, Michael::1, 242	Michael::3, 242
G.::5, 208	K.::5, 190	F.::5, 187
H.::5, 165	James::3, 160	, James::1, 159
N.::5, 156	, David::1, 155	David::3, 155
John::3, 153	, John::1, 152	, Robert::1, 150

Figure 3: Profiling and Listing the Patterns in the Data



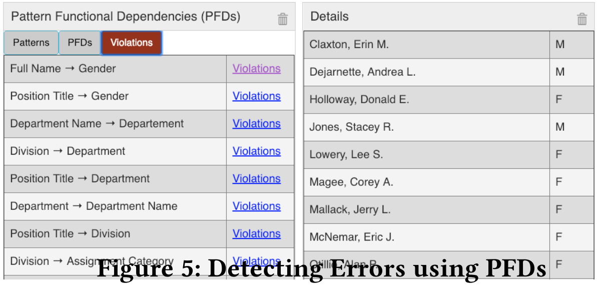
Pattern Functional Dependencies (PFDs)

Patterns | PFDs | Violations

Full Name → Gender	□	Tableau	Token combinations starting at the fourth token of the "Full Name" attribute determine the "Gender" values
Position Title → Gender	□	Tableau	Token combinations starting at the first token of the "Position Title" attribute determine the "Gender" values
Department Name → Departement	□	Tableau	Token combinations starting at the first token of the "Department Name" attribute determine the "Department" values
Division → Department	□	Tableau	Token combinations starting at the first token of the "Division" attribute determine the "Department" values
Position Title → Department	□	Tableau	Token combinations starting at the first token of the "Position Title" attribute determine the "Department" values

Figure 4: Displaying Discovered PFDs

PFD Discovery. An example of the extracted patterns is shown in Figure 3. The set of patterns are then used to extract the PFDs, and the PFDs that satisfy the minimum coverage will be reported. The user of ANMAT will be able to display the tableau of each dependency and confirm whether that discovered dependency is valid for the dataset at hand (Figure 4). The displayed patterns have the form "pattern::position, frequency", where the position represents the token number at which the combination of tokens that form the pattern start, assuming that the position of the first token is 0. The frequency represents the number of tuples that contain the pattern. When the patterns are extracted using n-grams, the position represents the position of the character at which the n-gram starts. Please note that n-grams are mainly used to extract patterns from attributes that contain single token which could be a code or ids.



Pattern Functional Dependencies (PFDs)

Patterns | PFDs | Violations

Full Name → Gender	Violations
Position Title → Gender	Violations
Department Name → Departement	Violations
Division → Department	Violations
Position Title → Department	Violations
Department → Department Name	Violations
Position Title → Division	Violations
Division → Assignment Category	Violations

Details

Claxton, Erin M.	M
Dejarnette, Andrea L.	M
Holloway, Donald E.	F
Jones, Stacey R.	M
Lowery, Lee S.	F
Magee, Corey A.	F
Mallack, Jerry L.	F
McNemar, Eric J.	F
Otillio, Alan P.	F

Figure 5: Detecting Errors using PFDs

Data	Dependendcy	Pattern Tableau	Errors
D_1	Phone Number → State	$\overline{850}\backslash\text{D}\{7\} \longrightarrow \text{FL}$ $\overline{607}\backslash\text{D}\{7\} \longrightarrow \text{NY}$ $\overline{404}\backslash\text{D}\{7\} \longrightarrow \text{GA}$ $\overline{217}\backslash\text{D}\{7\} \longrightarrow \text{IL}$ $\overline{860}\backslash\text{D}\{7\} \longrightarrow \text{CT}$	8505467600 | CA 6073771300 | PA 4048481918 | OK 2176163297 | TX 8602713444 | SC
D_2	Full Name → Gender	$\backslash\text{A}*,\backslash\text{\textvisiblespace}\overline{\text{Donald}}\backslash\text{A}*\longrightarrow\text{M}$ $\backslash\text{A}*,\backslash\text{\textvisiblespace}\overline{\text{Stacey}}\backslash\text{A}* \longrightarrow \text{F}$ $\backslash\text{A}*,\backslash\text{\textvisiblespace}\overline{\text{David}} \longrightarrow \text{M}$ $\backslash\text{A}*,\backslash\text{\textvisiblespace}\overline{\text{Jerry}}\backslash\text{A}* \longrightarrow \text{M}$ $\backslash\text{A}*,\backslash\text{\textvisiblespace}\overline{\text{Alan}}\backslash\text{A}* \longrightarrow \text{M}$	Holloway, Donald E. | F Jones, Stacey R. | M Kimbell, David | F Mallack, Jerry L. | F Otillio, Alan P. | F
D_5	ZIP ⟶ CITY	$\overline{6060}\backslash\text{D} \longrightarrow \text{Chicago}$ $\overline{6060}\backslash\text{D} \longrightarrow \text{Chicago}$ $\overline{6060}\backslash\text{D} \longrightarrow \text{Chicago}$	60601 | Chicag 60603-6263 | C 60601 | Chciago
D_5	ZIP ⟶ STATE	$\overline{60}\backslash\text{D}\{3\} \longrightarrow \text{IL}$ $\overline{95}\backslash\text{D}\{3\} \longrightarrow \text{CA}$	60603 | lL 95603 | MI

Table 3: Discovered PFDs and Detected Errors

Error Detection using Discovered PFDs. Based on the confirmed dependencies, ANMAT will run them through the corresponding columns and return all violations, which are highly likely to be erroneous values. Since easy validation of the reported errors increases data cleaning tools' usability, it is important for ANMAT to provide techniques to validate the errors. The user of ANMAT can display the violated rule(s) in the tableau and the full violating records to have more insights about the violations and confirm whether it is an error. Figure 5 show examples of reported violations for the dependency *Full Name* ⟶ *Gender*. More examples for errors discovered from different datasets are shown in Table 3.

REFERENCES

[1] S. Abiteboul, R. Hull, and V. Vianu. Foundations of databases. 1995.

[2] W. Fan, F. Geerts, X. Jia, and A. Kementsietsidis. Conditional functional dependencies for capturing data inconsistencies. *ACM Trans. Database Syst.*, 33(2):6:1–6:48, 2008.

[3] Z. Huang and Y. He. Auto-detect: Data-driven error detection in tables. In *SIGMOD*, pages 1377–1392, 2018.

[4] Z. Khayyat, I. F. Ilyas, A. Jindal, S. Madden, M. Ouzzani, J.-A. Quiane-Ruiz, P. Papotti, N. Tang, and S. Yin. BigDansing: a system for big data cleansing. In *SIGMOD*, 2015.

[5] A. A. Qahtan, A. K. Elmagarmid, R. C. Fernandez, M. Ouzzani, and N. Tang. FAHES: A robust disguised missing values detector. In *KDD*, pages 2100–2109, 2018.

[6] L. J. Stockmeyer and A. R. Meyer. Word problems requiring exponential time: Preliminary report. In *STOC*, pages 1–9, 1973.