Editing Rules: Discovery and Application to Data Cleaning

Thierno Diallo [#] [*], Jean-Marc Petit [#], and Sylvie Servigne [#]

[#] LIRIS UMR 5205 CNRS/Université de Lyon,
INSA de Lyon, bâtiment B. Pascal
20, Avenue Albert Einstein - 69622 Villeurbanne cedex
[*]Orchestra Networks SA
11 rue Scribe 75009 Paris-France
firstname.lastname@insa-lyon.fr

Abstract. Dirty data is a serious problem for businesses, leading to incorrect decision making, inefficient daily operations, and ultimately wasting both time and money. A variety of integrity constraints like Conditional Functional Dependencies (CFD) have been studied for data cleaning. Data repairing methods based on these constraints are strong to detect inconsistencies but are limited on how to correct data, worse they can even introduce new errors. Based on Master Data Management principles, a new class of data quality rules known as Editing Rules (eR) tells how to fix errors, pointing which attributes are wrong and what values they should take.
However, finding data quality rules is an expensive process that involves intensive manual efforts. In this paper, we develop pattern mining technics for discovering eRs from existing source relations (eventually dirty) with respect to master relations (supposed to be clean and accurate). In this setting, we propose a new semantic of eRs taking advantage of both source and master data. The problem turns out to be strongly related to the discovery of both CFD and one-to-one correspondences between sources and target attributes. We have proposed efficient technics to address these two subproblems.
We have implemented and evaluated our technics on real-life databases. Experiments show both the feasibility, the scalability and the robustness of our proposition.

1 Introduction

Poor data quality continues to be an important issue for companies. Erroneous, incomplete or duplicate data leads to bad and poor business decisions which cost a lot. There is an increased need for effective methods to improve data quality and to restore consistency.

A variety of integrity constraints have been studied for data cleaning from traditional functional and inclusion dependencies to their conditional extensions.

[6, 7, 14, 17]. These constraints help us to determine whether errors are present in the data but they fall short of telling us which attributes are concerned by

the error and moreover how to correct it. Worst, heuristic methods based on that constraints may introduce new errors when trying to repair the data. These limitations motivate [16] to define *Editing Rules*, a new class of data quality rules that tells how to fix errors, i.e. which attributes are wrong and what values they should take. Editing Rules (eRs) are boosted by the recent developpement of master data management (MDM [22, 11, 27, 29]) and then are defined in term of *master data*, a single repository of high-quality data that provides various applications with a synchronized, consistent view of its core business entities. This helps to fix and enrich dirty data using the corresponding values from master data.

However, for eRs to be effective in practice, it is necessary to have technics in place that can *automatically discover* such rules from both source relations and their corresponding master relations. Indeed it is often unrealistic to rely on human experts to design them by hand via an expensive and long manual process. This practical concern highlights the need for studing the *discovering problem* of eRs. On the one hand, it is worth nothing that any master relation s is assumed to contain only pure, accurate and curated data. So it makes sense to infer the "truth" from master data: they can be seen as an Oracle with respect to data quality. Cleary, the process of defining such master data from existing data sources is out of the scope of this paper [22]. On the other hand, we have to be more prudent and sceptic on data sources quality. Roughly speaking, the problem statement can be given as follows: Given an operational database $d = \{r_1, r_2, \ldots, r_n\}$ and a master database $m = \{s_1, s_2, \ldots, s_p\}$, infer all eRs to correct the database d with respect to the database m.
In the sequel, we assume the existence of both sources and master data. The former can be inaccurate and dirty, the latter is curated and clean.

The discovery problem is, however, highly non trivial. It is already hard for traditional dependencies and their corresponding conditional classes (CFDs and conditional inclusion dependencies) [9, 12, 21].

Our approach is based on the following process:

1. Finding one to one attribute mapping between source and master relation.
2. Infering rules from master relation.
3. Applying rules on input source relation.

Contributions: In this paper we define and provide solutions for dicovering eRs in databases provided with source data and master data. In particular we make the following contributions:

1. We introduce a new semantic of eRs allowing to precisely define the discovery problem as a pattern mining one.
2. We propose heuristics to apply eRs in a data cleaning process.
3. We present an experimental evaluation of our technics demonstrating their usefulness for discovering eRs. We finally evaluate the scalability and the robustness of our technics.

Related Works: This work finds similarities to constraint discovery and Association Rule (AR) mining. Plenty of contributions have been proposed for Functional Dependencies inference and AR mining [2, 26, 18, 20, 25], but less for CFDs mining [9, 15, 12]. In [9], authors propose a tool for data quality management which suggests possible rules and identify conform and non-conform records. They present effective algorithms for discovering CFDs and dirty values in a data instance, but the CFDs discovered may contain redundant patterns. In [15], authors proposed three methods to discover CFDs. The first one is called CFDMiner which mines only constant CFDs i.e. CFDs with constant patterns only. CFDMiner is based on technics for mining closed itemsets [26]. The two other ones, called CTANE and FastCFD, were developed for general (non constant) CFDs discovery. CTANE and FastCFD are respectively extensions of well known algorithms TANE [18] and FastFD [30] for mining FDs. In [12] authors propose a powerfull method for mining frequent constant CFDs.
The problem we address is also related to mapping function discovery between source and target schemas. Many contributions have been proposed in this setting. For example authors in [4], based on INclusion Dependencies (INDs), present the SPIDER algorithm that efficiently find all the INDs in a given relation. Authors in [19] construct a dependency graph as a measure of the dependency between attributes. Then they find matching node pairs in the dependency graphs by running a graph matching algorithm. In [31], authors propose a robust algorithm for discovering single-column and multi-column foreign keys, which can be used as a mapping algorithm for attributes. In addition many technics have been proposed to identify one-to-one correspondences in different domains such as schema matching [5, 28], ontology alignments [13] or understanding logical database design [21].
Since introduction of eRs by [16], as far as we know, no contributions have been made for the discovery problem of eRs.

Paper Organization: This paper is organized as follows. In section 2, we give preliminary definitions. In section 3 we describe the problem statement. We give details of our solutions for the eRs discovery problem in section 4. In section 5 we describe application of eRs to repair data. Section 6 presents an experimental evaluation of our methods and section 7 finally concludes the paper.

2 Preliminaries

We shall use classical database notions (e.g. [1], CFDs and INDs terminologies [7, 14, 17, 12, 21]). The relation symbol is generally denoted by R and the relation schema of R is denoted by $sch(R)$. When clear from context, we shall use R instead of $sch(R)$. Each attribute A has a domain, denoted by $DOM(A)$. A relation is a set of tuples and the projection of a tuple t on an attribute set X is denoted by $t[X]$. Given a relation r over R and $A \in R$, active domain of A in r is denoted by $ADOM(A, r)$. The active domain of r is noted $ADOM(r)$. Letters from the beginning of the alphabet $(A, B, C, \ldots)$ shall represent single

attribute whereas letters from the end of the alphabet $(X, Y, Z, \ldots)$ attribute sets. For convenience, XY will refer to as $X \cup Y$.

2.1 Conditional Functional Dependencies (CFDs)

The reader familiar with CFDs may skip this section. CFDs have been recently introduced in the context of data cleaning [7]. They can be seen as an unification of Functional Dependencies (FD) and Association Rules (AR) since they allow to mix attributes and attribute/values in dependencies.

We now consider a relation schema R, the syntax of a CFD is given as follows:

Definition 1. ***Syntax:*** *A Conditional Functional Dependency (CFD) ρ on R is a pair $(X \rightarrow Y, T_p)$ where (1) $XY \subseteq R$, (2) $X \rightarrow Y$ a standard Functional Dependency (FD) and (3) T_p is a pattern tableau with attributes in R.*

For each $A \in R$ and for each pattern tuple $t_p \in T_p$, $t_p[A]$ is either a constant in $DOM(A)$, or an 'unnamed variable' denoted by '_', or an empty variable denoted by '$*$' which indicates that the corresponding attribute does not contribute to the pattern (i.e. $A \notin XY$).

The semantics of a CFD extends the semantics of FD with mainly the notion of matching tuples.

Let r be a relation over R, $X \subseteq R$ and T_p a pattern tableau over R. A tuple $t \in r$ *matches* a tuple $t_p \in T_p$ over X, denoted by $t[X] \asymp t_p[X]$, iff for each attribute $A \in X$, either $t[A] = t_p[A]$, or $t_p[A] =$'_', or $t_p[A] =$'$*$'.

Definition 2. ***Semantic:*** *Let r be a relation over R and $\rho = (X \rightarrow Y, T)$ a CFD with $XY \subseteq R$. We say that r satisfies ρ, denoted by $r \models \rho$, iff for all t_i,$t_j \in r$ and for all $t_p \in T$, if $t_i[X] = t_j[X] \asymp t_p[X]$ then $t_i[Y] = t_j[Y] \asymp t_p[Y]$.*

That is, if $t_i[X]$ and $= t_j[X]$ are equal and in addition, they both match the pattern $t_p[X]$, then $t_i[Y]$ and $= t_j[Y]$ must also be equal to each other and both match the pattern $t_p[Y]$.

Example 1. Let r_0 be a relation defined over $ABCD$ (Figure 1). We can say $r_0 \models (AC \rightarrow D, (2, *, 0 \parallel 1))$.

r_0	A	B	C	D
t_1 :	0	1	0	2
t_2 :	0	1	3	2
t_3 :	0	0	0	1
t_4 :	**2**	2	**0**	**1**
t_5 :	**2**	1	**0**	**1**

Fig. 1. A Toy relation r_0

We say that r satisfies a set Σ of CFD over R, denoted by $r \models \Sigma$ if $r \models \rho$ for each CFD $\rho \in \Sigma$. A CFD $(X \to Y, T_p)$ is in the *normal form* [14], when $|Y| = 1$ and $|T_p| = 1$. So a normalized CFD has a single attribute on the right-hand side and its pattern tableau has only one single tuple. In the sequel we consider CFDs in their normal form, unless stated otherwise. A CFD $(X \to A, t_p)$ is called:

- a *constant CFD* if $t_p[XA]$ consists of constants only, i.e. $t_p[A]$ is a constant and $t_p[B]$ is also a constant for all $B \in X$.
- a *variable CFD* if the right hand side of its pattern tuple is the unnamed variable '_', i.e. $t_p[A] =$ '_', the left-hand side involving either constants or '_'.

Compare to FDs, note that a single tuple relation may violate a CFD. It may occur when the pattern tableau has at least one row with at least one constant on the right-hand side. Given a relation, the satisfaction of a CFD has to be checked with both every single tuple and every couple of tuples. More formally, we get:
r violates a CFD $\rho = (X \to A, T)$, denoted by $r \not\models \rho$, iff

- there exists a tuple $t \in r$ and a pattern tuple $t_p \in T$ such that $t[X] \asymp t_p[X]$ and $t[A] \not\asymp t_p[A]$. This may occur when $t_p[A]$ is a constant.
- or there exists $t_i, t_j \in r$ and a pattern tuple $t_p \in T$ such that $t_i[X] = t_j[X] \asymp t_p[X]$ and $t_i[A] \neq t_j[A]$. We can assume that $t_p[A] =$ '_' since otherwise the violation is already covered by the single tuple violation.

Example 2. Figure 1, $r_0 \not\models (A \to D, (0, *, * \parallel 2))$. The tuple t_3 violates the constraint, indeed $t_3[A] = 0$ and $t_3[D] \neq 2$.

2.2 Editing Rules (eRs)

eRs syntax [16] is slightly rephrased in the following definition.

Definition 3. ***Syntax:*** *Let R and S be two relation symbols. R is the relation source symbol and S is the target master one. Let X (resp. Y) be a list of distinct attributes from $sch(R)$ (resp. $sch(S)$), $A \in sch(R) \setminus X, B \in sch(S)$ and $Z \subseteq sch(R)$. An eR φ over (R, S) is of the form φ :*
$((X, Y) \to (A, B), t_p[Z])$ where $t_p[Z]$ is a pattern tuple over R.

Let $r \in sch(R)$ and $s \in sch(S)$ be respectively a source and a master relation. The semantics of eRs have been defined with respect to the insertion of a tuple t in r [16]. The idea is to "correct" r using values of s with respect to pattern selection applying on r.

Definition 4. ***Semantic:*** *An eR $\varphi = ((X, Y) \to (A, B), t_p[Z])$, with respect to a master tuple $t_m \in s$, is applied to $t \in r$ to obtain t' if:*

1. *$t[Z]$ matches $t_p[Z]$*
2. *$t[X] = t_m[Y]$*

r	FN	LN	AC	phn	type	str	city	zip	item
t_1 :	Bob	Brady	020	079172485	2	501 Elm St.	Edi	EH7 4AH	CD
t_2 :	Robert	Brady	131	6884563	1	null	Lnd	null	CD
t_3 :	Robert	Brady	020	6884563	1	null	null	EH7 4AH	DVD
t_4 :	Mary	Burn	029	9978543	1	null	Cad	null	BOOK

s	FN	LN	AC	Hphn	Mphn	str	city	zip	DOB	gender
s_1 :	Robert	Brady	131	6884563	079172485	51 Elm Row	Edi	EH7 4AH	11/11/55	M
s_2 :	Mark	Smith	020	6884563	075568485	20 Baker St.	Lnd	NW1 6XE	25/12/67	M

u	ItemId	Item	DOF	Price
u_1 :	701B	Book	04/03/10	45
u_2 :	017A	CD	11/10/10	11
u_3 :	904A	DVD	25/08/10	16

Fig. 2. source relation r and master database $\{s, u\}$

3. t' is obtained by the update $t[A] := t_m[B]$

That is, if t matches t_p and if $t[X]$ equals $t_m[Y]$, then $t_m[B]$ is assigned to $t[A]$.

Example 3. Let us consider the tuples t_1 and s_1 in Figure 2. t_1 is in a source relation and s_1 is in a master relation. It is known that if Area Code (AC) is 020, city should be London (Lnd), and when AC is 131, city must be Edinburgh (Edi). The tuple t_1 is inconsistent: $t_1[AC] = 020$ but $t_1[city] = Edi$. So, an example of eR that correct such an inconsistency is $((zip, zip) \rightarrow ((AC), (AC)), t_{p1} = ())$. In that case, $t_1[AC]$ is changed to (131) taken from s_1.
Similarly $t_1[FN] = Bob$ may be corrected using the right name $s_1[FN] = Robert$ with respect to the eR $((phn, Mphn) \rightarrow ((FN), (FN)), t_{p2}[type] = (2))$. That is, if $t[type] = 2$ (indicating mobile phone) and if there is a master tuple s with $s[Mphn] = t[phn]$, then $t[FN] := s[FN]$.

3 Problem Statement

The problem we are interested in can be stated as follows: given an instance r from a source database d and an instance s from a master database m, it is to find all applicable eRs on r with respect to s.
However, in a data mining setting, the previous semantic of eRs (Definition 4) needs to be properly extended since it just involves tuple insertion. We need to define a semantic to identify rules.
In the sequel, to alleviate notations the master database is reduced to a single master relation. Taking into account a set of relations instead of a single relation is doable easily.

Preliminary, we introduce a general schema mapping function f between attributes of source relation and master relation. This function is based on the following requirements:

– Each attribute of a source relation should have a mapping attribute in a master relation, and only one.
– If two attributes A, B of a source relation have the same mapping attribute in a master relation, then $A = B$.

Let $sch(R)$ and $sch(S)$ be respectively the source schema relation and the master one.

Definition 5. *A mapping function f from R to S, is defined as a total and injective function; $f : sche(R) \rightarrow sche(S)$*

By extension, for $X \subseteq sch(R)$ we note: $f(X) = \bigcup_{A \in X} \{f(A)\}$. The new definition of eRs semantic can now be given.

Definition 6. ***New semantic:*** *Let $\varphi = ((X, Y) \rightarrow (A, B), t_p[Z])$ be an eR over (R, S), r a source relation over R and s a master relation over S.*

We say that φ is satisfied in (r, s) with respect to f, denoted by $(r, s) \models_f \varphi$, if:

1. $f(X) = Y$
2. $f(A) = B$
3. $f(Z) = Z'$
4. $s \models Y \rightarrow B, t_p[Z']$

Example 4. Let $\varphi = ((zip, zip) \rightarrow (AC, AC), t_p = (EH74AH))$ be an eR defined on r, s of Figure 2. We can say that φ is satisfied in (r, s) with respect to the identity function. Indeed, attributes are the same (i.e $zip = zip$) and $s \models zip \rightarrow AC, t_p = (EH74AH)$.

The problem of Discovering eRs (denoted by **DER**) is defined as follows:

> Given a source relation r, a master relation s and a mapping function f, find a cover of all eRs satisfied in (r, s) with respect to f.

4 Mining Editing Rules

In order to solve the DER problem, we have to deal with the following steps:

1. Specifying a mapping function f and discovering the mapping between attributes related to f.
2. Discovering a cover of CFDs satisfied in s,
3. Propagating CFDs from s to r using f: for each discovered CFDs $(Y \rightarrow B, t_p[Z'])$, generating an eR $((f^{-1}(Y), Y) \rightarrow (f^{-1}(B), B), f^{-1}(t_p[Z']))$.

In the sequel, we first consider the simplest case: the input source relation and master relation are defined over the same relation schema, i.e. the mapping function f is the identity function. Then we address the general case when relation schemas are different.

4.1 Same Schemas on source and master relation

In this approach, let us assume that the schema of R is equal to the schema of S. The syntax definition 3 of eRs is adapted accordingly as follows:.

Definition 7. ***New Syntax:*** *Let R and S be two relation symbols with $sch(R) = sch(S)$. Let X be a list of distinct attributes from $sch(R)$, $A \in sch(R) \setminus X$ and $Z \subseteq sch(R)$. An eR φ over (R, S) is of the form $\varphi : (X \rightarrow A, t_p[Z])$ where $t_p[Z]$ is a pattern tuple over R.*

This definition is closer to CFD definition. The CFDs discovery problem has been already studied in [9, 12, 15]. In [12] we proposed efficient technics to discover all constant CFDs satisfied by a given relation. We focus on two types of technics inherited from FD inference: the first one extends the notion of agree sets and the second one extends the notions of non-redundant sets, closure and quasi-closure. The DER problem can be resolved as follows:

1. Discovering a cover of CFDs satisfied in s,
2. For each discovered CFDs $(X \rightarrow A, t_p[Z])$, generating the eR $((X, X) \rightarrow (A, A), t_p[Z])$.

4.2 Different Schemas on source and master relation

In this case, it is necessary to find out, through a mapping function, correspondences between attributes of master relation and source relation. Different technics can be used to specify such a mapping function. For instance, one may relay on attribute naming assumption not based on instances or, ontology alignment [13], discovery of inclusion dependencies (INDs) [21, 4], both based on instances.

For a given correspondence, values should be as much as possible "similar", i.e. some forms of INDs should exist between them. Clearly, due to the presence of errors in source relations and due to the high quality data of the master relation, the definition of such a correspondence should be flexible, which is indeed possible with approximative unary INDs.

4.2.1 From source and master relations to INDs

We define the mapping function f with respect to unary INDs from the source relation r to the master relation s. Efficient technics have been proposed to discover such unary IND satisfied in a given database, among them we quote [21, 4].

Let A and B be respectively single attributes. The unary IND $A \subseteq B$ means all values of attribute A are included in the bag of values of attribute B. An IND candidate is satisfied when it fulfills the IND definition. For example in Figure 2, we have: $Hphn \subseteq phn$.

We want to identify all unary INDs based on both source and master relations. To that end, we introduce a preprocessing of relations to be as much as closer to association rule and transaction syntax. We call such preprocessing *condensed representation*.

Given a common relation r over a schema $sch(R)$ the *condensed representation* of r, denoted by $CR(r)$, is defined by:
$CR(r) = \{(v, X) | v \in ADOM(r), X = \bigcup\{A \in sch(R) | \exists t \in r, t[A] = v\}\}$. For a set of relations, $CR(r_1, \ldots, r_n) = \bigcup_{i=1..n} CR(r_i)$.

Example 5. The condensed representation $CR(r_0)$ of relation r_0 (Figure 1 recalled below) is given as follows:

r_0	A	B	C	D
t_1 :	0	1	0	2
t_2 :	0	1	3	2
t_3 :	0	0	0	1
t_4 :	**2**	2	**0**	**1**
t_5 :	**2**	1	**0**	**1**

$CR(r_0)$:

0 ‖ A B C
1 ‖ B D
2 ‖ A B D
3 ‖ C

For convenience concerning the condensed representation, we use the following notations:

- $CR(r).val = \{v | (v, X) \in CR(r)\}$
- $CR(r).v.atts = \{X | (v, X) \in CR(r)\}$

The *support* of an attribute set $X \subseteq R$ in $CR(r)$, denoted by $sup(X, CR(r))$, is defined by:

$$sup(X, CR(r)) = |\{(v, Y) \in CR(r) | X \subseteq Y\}|$$

The *closure* of an attribute $A \in sch(R)$ with respect to $CR(r)$, denoted by $A^+_{CR(r)}$, is defined as:

$$A^+_{CR(r)} = \bigcap_{(v,X) \in CR(r)} \{X | A \in X\}$$

Property 1. Let r be a relation over R and A, B attributes of R.
$r \models A \subseteq B \iff B \in A^+_{CR(r)} \iff sup(\{A, B\}, CR(r)) = sup(\{A\}, CR(r))$

Proof. $r \models R[A] \subseteq S[B]$.
For all $v \in \pi_A(r)$, $\exists t \in r$ such that $v = t[B]$. So $v \in \pi_A(r) \iff v \in \pi_B(r)$.
$\forall (v, X) \in CR(r)$, $A \in X \iff B \in X$.
So $B \in \bigcap_{(v,X) \in CR(r)} \{X | A \in X\}$.
Finally $B \in A^+_{CR(r)}$

A consequence of property 1 is on the computation of the closure: it is no longer necessary to compute the intersection between attributes to extract INDs.

To identify all approximative unary INDs it is necessary to use a natural error measure, called g'_3 [21]. Let r be a source over schema R and let s be a master relation over schema S.

$$g'_3(R[X] \subseteq S[Y], \{r, s\}) = 1 - \frac{max\{|\pi_X(r')| : r' \subseteq r, \{r, s\} \models R[X] \subseteq S[Y]\}}{|\pi_X(r)|}$$

Example 6. Figure 2: $g'_3(R[AC] \subseteq S[AC], \{r, s\}) = 1 - (2/3) = 1/3$

Correspondence between attributes can be characterized using such measure.

Definition 8. *Let $A \in sch(R), B \in sch(S)$ and ϵ a $[0, 1]$-threshold value.*

We say that A corresponds to B in r, s with respect to ϵ, denoted by $\{r, s\} \models_\epsilon corr(A, B)$, if

$$g'3(R[A] \subseteq S[B], \{r, s\}) \leq \epsilon$$

By extension, we say that X corresponds to Y wrt ϵ if $\forall A \in X, \exists B \in Y$ such that $\{r, s\} \models_\epsilon corr(A, B)$

The error measure can be also defined using the support of attribute sets as follows:

Definition 9. *Let r be a relation over R and $A, B \in R$.*

$$error(A \subseteq B) = \frac{sup(\{A, B\})}{sup(\{A\})}$$

Property 2. Let r be a relation over R and $A, B \in R$.

$$error(A \subseteq B) = g'_3(A \subseteq B)$$

Thanks to definition 9 and property 2, we just need to compute the support of every single attribute (item of size 1) and the support of every couple of generated candidates attributes (CandidateGeneration) for A and B (itemsets of size 2).
Therefore, we use the APRIORI [3] algorithm until level 2 without any threshold.

Algorithm 1 ScalableClosure

Require: A condensed representation CR of r and s
Ensure: F_1, F_2, itemsets and their supports of size 1 and 2 respectively
1: $F_1 = \{(A, support(A)) | A \in R\}$
2: $C_2 = CandidateGeneration(F1)$
3: **for all** $(v, X) \in CR(r)$ **do**
4: $F_2 = subset(C_2, X)$ – Return the subset of C_2 containing X
5: **for all** $e \in F_2$ **do**
6: $support(e) += 1$
7: **end for**
8: **end for**
9: **return** F_1, F_2

4.2.2 From approximative unary INDs to a mapping function f

Given the source relation, the master relation and a ϵ threshold, our goal is find mapping between attributes based on approximative unary INDs. A mapping function can be defined in this setting as described in Algorithm 2.

Algorithm 2 (SR2MR) Mapping from Source Relation to Master Relation

Require: a source relation r over R, a master relation s over S
Ensure: A mapping function f from R to S
1: $CR = Preprocessing(r, s)$;
2: $(F_1, F_2) = scalableClosure(CR)$;
3: **for all** $A \in R$ **do**
4: $((B, \epsilon), F_2) = FindBest(F_1, F_2, A)$;
5: **while** $g'_3(A \subseteq B) \leq \epsilon$ **do**
6: $\epsilon = \epsilon + 0.05$
7: **end while**
8: $f(A) = B$
9: **end for**
10: **return** f

The first step of Preprocessing(r,s) computes for both relations the condensed representation described in example 5. Itemsets $F1$, $F2$ respectively of size 1 and 2 and their supports are generated thanks to Algorithm 1. And for each attribute A in R a corresponding attribute in S is mined by *FindBest* procedure with respect to a mapping function f. When there is no corresponding attribute, the ϵ threshold is decreased until one is found through approximative unary IND. When many approximative unary INDs are concerned, the one with the biggest cardinality is choosen. This procedure is presented on Algorithm 3 where the symbol $\perp$ refers to the default attribute.

Algorithm 3 FindBest

Require: F_1, F_2 itemsets of size 1 and 2 respectively and their supports , $A \in R$
Ensure: $B \in S$, a mapping attribute for A,
F_2, remaining (itemsets,support) of size 2
1: Let $(A, v) \in F_1$;
2: Let $E = \{(A_iA_j, v) \in F_2 | A = A_i \text{ or } A = A_j\}$
3: **if** $\exists (AB, v) \in E$ such that for all $(X, v') \in E, v \geq v'$ **then**
4: Remove all occurrences of B in F_2
5: **return** $((B, \frac{v'}{v}), F_2)$
6: **else**
7: **return** $((\perp, 1), F_2)$
8: **end if**

Finally Algorithm 2 outputs a total and injective function f whenever $S \supseteq R$.

4.2.3 Mining Editing Rules

The process of discovering eRs begins by computing the condensed representation of all the relations. Then we define the mapping function in order to find one to one correspondences between attributes. Finally, with respect to the set

of satisfied CFDs in master data, we build the corresponding eRs to solve DER problem.

Algorithm 4 Discovery of Editing Rules

Require: r a source relation, s a master relation, Σ the set of CFDs satisfied in s, ϵ a threshold.
Ensure: eRs for r with respect to s

```
1: res = ∅;
2: f = SR2MR(r, s);
3: for all A ∈ R do
4:    Let (s.B, err) = f(A);
5:    CFD = {cfd ∈ Σ| cfd defined over s }
6:    for all X → A, t_p[Z] ∈ CFD do
7:       if g(A ∪ X ∪ Z) ∈ R then
8:          if ∀B ∈ (A ∪ X ∪ Z) such that f(g(B)).err ≤ ε then
9:             res+ = (f^-1(X), X) → f^-1(A), A), t_p[f^-1(Z)]
10:         end if
11:      end if
12:   end for
13: end for
14: return res
```

5 Applying Editing Rules

Applying eRs is actually a process of repairing data [16]. Repairing data using contraints is challenging specially for CFDs [7].

5.1 Main problems of data repairing with CFDs

5.1.1 CFDs violation

CFDs violation may be impossible to solve. To see this, consider a schema R with $\{A, B, C\}$ attributes, an instance I of R consisting of (a_1, b_1, c_1) and (a_1, b_2, c_2), and a set $\Sigma = \{(A \rightarrow B, (\_, \_)), (C \rightarrow B, ((c_1, b_1), (c_2, b_2)))\}$ of CFDs. Then $I \not\models \Sigma$ and moreover concerned CFDs are in conflict, any repair of I has to modify the values of some attributes in the left hand side of the CFD.

5.1.2 Application order

The correction of dirty tuples can cause some rules no more applicable. In this setting the order in which they are applied is important in order to maximize the number of rules used in practice to correct data. This issue is illustrated in the following with respect to CFDs as example.
Let r_1 be a relation defined on the following schema $ABCDE$, with a set of CFDs.

r_1	A	B	C	D	E
t_1 :	0	0	1	2	3
t_2 :	0	0	1	4	5

$$\varrho_1 = (DC \rightarrow E, (*, *, 2, 2 \parallel 5))$$
$$\varrho_2 = (AB \rightarrow C, (0, 0 \parallel 2, *, *))$$
$$\varrho_3 = (CD \rightarrow E, (*, *, 1, 2 \parallel 5))$$
$$\varrho_4 = (CD \rightarrow E, (*, *, 1, 4 \parallel 5))$$

Fig. 3. a relation r_1 over $ABCDE$ and a set of CFDs

When the set of CFDs is applied in the order they appear on Figure 3, only ϱ_2 is applied. There are no tuples matching pattern $D = 2, C = 2$ of ϱ_1 and they are no tuples matching pattern of ϱ_3 and ϱ_4 because ϱ_2 changed the value of $t[C]$ from 1 to 2.

In another hand, if we apply the set in the following order ϱ_4, ϱ_3, ϱ_2 and ϱ_1 all of them are actually applied. This example highlights the importance of the order in which rules are applied. Therefore it is necessary to develop heuristics that maximize the number of rules used in practice.

5.2 Heuristics for data repairing

In the sequel we recall the runing example of [16] with some modifications to have the same relation schema, i.e. *Hphn*, *Mphn*, *DOB* and *gender* attributes are removed from master relation schema S. Similary *phn*, *type* and *item* attributes are removed from R. Thus S and R schemas are now equal and contain FN, LN, AC, *str*, *city* and *zip*.

source : r	FN	LN	AC	str	city	zip
t_1	Bob	Brady	020	501 Elm St.	Edi	EH7 4AH
t_2	Robert	Brady	131	null	Lnd	null
t_3	Robert	Brady	020	null	null	EH7 4AH
t_4	Mary	Burn	029	null	Cad	null

master : s	FN	LN	AC	str	city	zip
s_1	Robert	Brady	131	51 Elm Row	Edi	EH7 4AH
s_2	Mark	Smith	020	20 Baker St.	Lnd	NW1 6XE

Fig. 4. Example from [16] modified.

Let us consider the example of Figure 4. The set of CFDs satisfied in the master relation s includes:
$s \models \{(Zip \rightarrow AC, (EH74AH, 131)), (Zip \rightarrow str, (EH74AH, 51ElmRow))\}$
Then two eRs can be generated:

– $((Zip, Zip) \rightarrow (AC, AC), (EH74AH, 131))$
– $((Zip, Zip) \rightarrow (str, str), (EH74AH, 51ElmRow))$

Therefore, the tuple t_1 in the source relation can be updated: $t_1[AC, str]$ is changed to (131, 51 Elm Row).
Once eRs are discovered, we can now apply them to clean data.

5.2.1 Heuristic H_0: Baseline

The use of eRs in a basic data cleaning process is described in Algorithm 5. Discovered eRs Σ are used to correct dirty relation r. The number of corrected tuples (cpt) is kept to evaluate the accuracy of the process.

Algorithm 5 Heuristic H_0:Baseline

Require: r, Σ
Ensure: A new relation $r' \models \Sigma$,
cpt, the number of corrected tuples of r.
1: $cpt = 0$;
2: $r' = r$;
3: **for all** $t \in r'$ **do**
4: **for all** $(X, Y) \rightarrow (A, B), t_p[Z] \in \Sigma$ such that $t \asymp t_p[Z]$ **do**
5: **if** $t[X] \asymp t_p[Z]$ **then**
6: $t[A] := t[B]$
7: $cpt++$
8: **end if**
9: **end for**
10: **end for**
11: **return** r', cpt

This strategy recalls the repairing algorithm *BatchRepair* provided by [10] which addresses these issues. We simplify the procedure by correcting dirty values using master relation values carried by eRs.

5.2.2 Heuristic H_0^*: Baseline-Recall

In practice, Baseline (Algorithm 5) is more efficient when repeated (for the same set of rules). The heuristic H_0^* called Baseline-Recall iterates over rules. Therefore a rule not applied at a given iteration step i can be applied at the next iteration step $i + 1$. For example, on Figure 3 at a first step we apply the set of rules ϱ_1, ϱ_2, ϱ_3 and ϱ_4 using heuristic H_0. Only ϱ_2 is applied. As a consequence, $t[C]$ is changed from 1 to 2. Using heuristic H_0^* ensure a second iteration and then ϱ_1 can be now applied because t_2 matches the pattern of ϱ_1. Since we have no garantee of terminaison, we have set a maximum value of iteration to 100.

5.2.3 Heuristic H_1: Left-Hand-Side-Length-Sorted

We previously studied the importance of the order in which rules are applied in order to maximize the number of rules used in practice. Several technics can be used. For example, we can sort the rules with respect to the size (in term of number of attributes) of their left hand side. The intuition is to apply rules with less selectivity.

The Recall strategy can be also applied to H_1 to obtain Left-Hand-Side-Length-Sorted-Recall heuristic (H_1^*). All these alternatives are experimentally verified in the next section.

6 Experimental Study

We ran experiments to determine the effectiveness of our proposed technics. We report our results using real dataset and provide examples demonstrating the quality of our discovered eRs. In particular we use HOSP [1] dataset maintained by the U.S. Departement of Health & Human Services, and come from hospitals that have agreed to submit quality information for Hospital Compare to make it public.

Our experiments were run using a machine with an Intel Core 2 Duo (2.4GHz) CPU and 4GB of memory.

6.1 Discovering Editing Rules

Our experimental study has been set up in the same condition of [16]. We use the same HOSP data and use the same three tables HOSP, HOSP_MSR_XWLK and STATE_MSR_AVG. HOSP records the hospital information including provider number (id), hospital name (hName), phone number (phn), state (ST), zip code (ZIP) and address. HOSP_MSR_XWLK records the score of each measurement on each hospital in HOSP, e.g. measure name (mName), measure code (mCode) and the score of the measurement for this hospital (Score). STATE_MSR_AVG records the average score of each measurement on hospitals in all US states, e.g. state (ST), mName and state average (sAvg) the average score of all hospitals in this state.
First we present mapping function extraction, then we experiment eRs mining.

6.1.1 Eliciting mapping function w.r.t. unary INDs

We have used Apriori algorithm [3] for mining mapping function from source relation to master database. Many implementations exits, we have chosen the open source implementation of [8] which ensure a polynomial behaviour in term of response time. This step being easy we do not provide experimental results (less than 10s).

[1] http://www.hospitalcompare.hhs.gov

6.1.2 CFDs inference

In [16], authors manually designed for HOSP data 37 eRs in total, obtained by a carefull analysis[2]. Five important ones cited by [16] are:
$\varphi_1 = ((zip, zip) \rightarrow (ST, ST), t_{p1}[zip] = ());$
$\varphi_2 = ((phn, phn) \rightarrow (zip, zip), t_{p2}[phn] = ());$
$\varphi_3 = ((mCode, ST), (mCode, ST)) \rightarrow (sAvg, sAvg), t_{p3} = ());$
$\varphi_4 = ((id, mCode), (id, mCode)) \rightarrow (Score, Score), t_{p4} = ());$
$\varphi_5 = (id, id) \rightarrow (hName, hName), t_{p5} = ());$

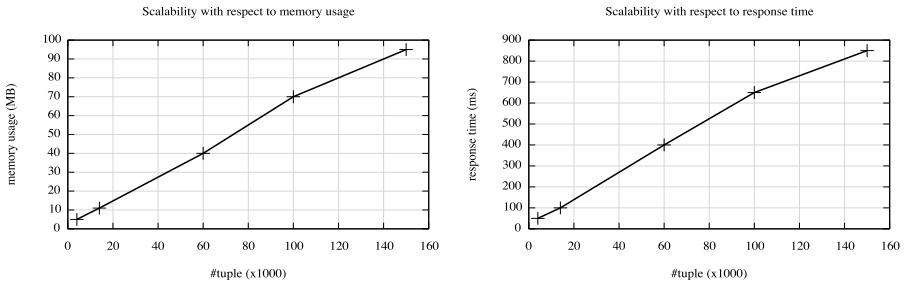

Fig. 5. Scalability w.r.t. response time and memory usage

We have run the *Cfun* [3] open source implementation of constant CFDs discovery [12] and we have been able to recover all eRs listed by [16].
For example the eR $\varphi_5 = (id, id) \rightarrow (hName, hName), t_{p5} = ())$ is equivalent to the set of constant CFDs in the form $id \rightarrow hName$.

All constant CFDs satisfied by the master relation have been extracted. Figure 5 shows the scalability with respect to response time and memory usage. The algorithm scales lineary both for execution time and memory usage [12].

6.2 Data Repairing

The experimental protocol is defined as follows:

1. Duplicating experimental relation r to obtain r'.
2. Introducing noise into r'. Null values are introduce into the relation. The noise rate of the relation is defined as the ratio of (number of noisy tuples) / (number of total tuples).
3. Discovering rules Σ from r.
4. Applying Σ to r'.

[2] Note to the referee: At the time of the submission of our paper, authors of [16] do not disclosed all their eRs, only 5 of them are publicly available
[3] http://pageperso.lif.univ-mrs.fr/ noel.novelli/CFDProject/

We evaluate the scalability of the repair process. Figure 6 shows that the process scales very well for source relation sizing from 4000 to 160000 tuples and for noise rate of 5 and 10 percent.

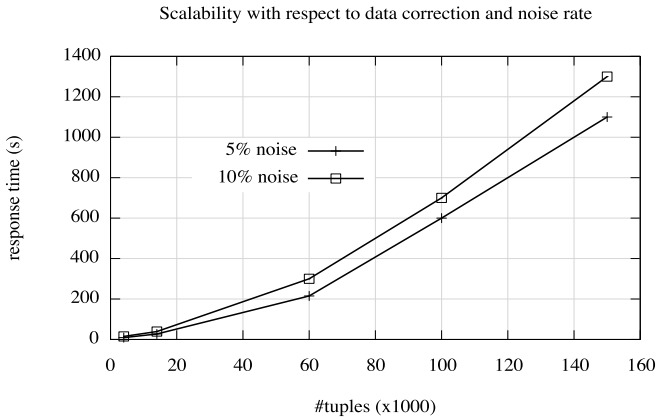

Fig. 6. Scalability w.r.t. data correction and noise rate

We evaluate the quality of the repairing process embedded in the discovered eRs. We show the accuracy in terms of pourcentage of errors corrected, defined as the ratio of (total number of corrected tuples) / (total number of noisy tuples). The source relation still may contain noises that are not fixed [10].

In the experiment of Figure 7 we have varied noise rate from 1 to 10 percent and set the total number of tuples to 100 000. The repair quality decreases when the noise rate increases, but it remains superior to 82 percent for all strategies. Heuristics are quiet equivalent when inconsistencies are in low rate, less than 2% of noise. Baseline (H_0) implements the Algorithm 5 which apply rules on the same order they are discovered and once. When the process is repeated the total of effective rules applied increase, therefore "Baseline-Recall" (H_0^*) increases the accuracy rate. The benefit increases more when rules are sorted with respect to the length of left hand sides. From H_0 to "Left-Hand-Side-Length-Sorted" H_1 heuristic, the accuracy grows from 82 to 90% for a noise rate of 10%. The combination of sorting with respect to left hand sides length and iteration (H_1^*) gives the best accuracy rate. We iterate hundred times for Recall heuristics.

7 Conclusion

Editing Rules are a new class of data quality rules boosted by the emergence of master data both in industry [11, 27, 29] and academy [16]. In this paper we propose a new semantic of Editing Rules in order to be able to infer them from

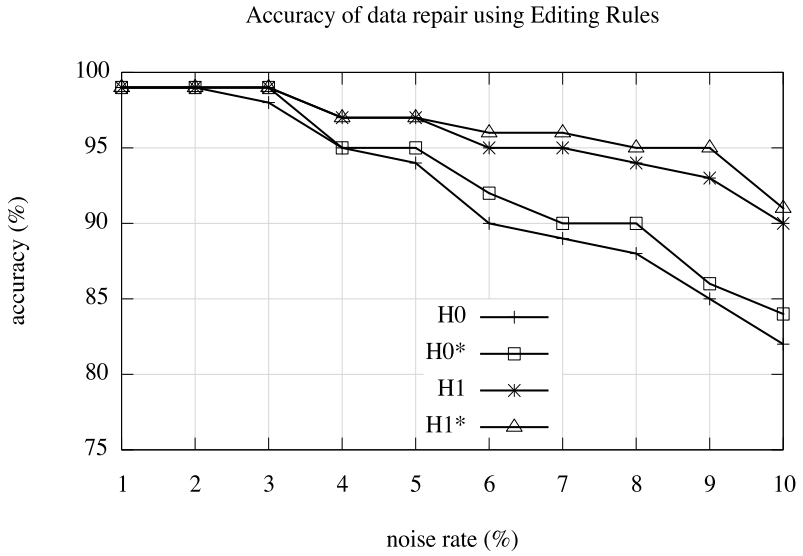

Fig. 7. Accuracy of data repair using Editing Rules

existing source database and a corresponding master database. Based on this new semantic, we have proposed a mining process in 3 steps:

- Eliciting one to one correspondences between attributes of a source relation and attributes of the master database.
- Mining CFDs in the master relations.
- Building Editing Rules.

Then we tackled the problem of data repairing from our discovered Editing Rules. We have proposed a few heuristics for that. Finally, we have developed the algorithms and ran experiments. Results obtained have shown that our approach scale well in the size of the database and provides good repairing results.

Many perspectives exit, for instance the inference of many to many correspondences instead of one to ones. The main problem is then to propagate the CFDs into the source data with the "right" attribute. We quote also that the notion of equivalence classes explored in [6] could be interesting in our setting. We also plan to extend our propositions to deal with many source relations in input.

References

1. S. Abiteboul, R. Hull, and V. Vianu. *Fondations of Databases*. Vuibert, 2000.
2. R. Agrawal, T. Imielinski, and A. N. Swami. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Washington, D.C., May 26-28, 1993*, pages 207–216. ACM Press, 1993.
3. R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proceedings of the 1994 VLDB International Conference, 1994*. VLDB Press, 1994.
4. J. Bauckmann, U. Leser, F. Naumann, and V. Tietz. Efficiently detecting inclusion dependencies. In *ICDE*, pages 1448–1450, 2007.
5. Z. Bellahsene, A. Bonifati, and E. Rahm, editors. *Schema Matching and Mapping*. Springer, 2011.
6. P. Bohannon, W. Fan, and M. Flaster. A cost-based model and effective heuristic for repairing constraints by value modification. In *In ACM SIGMOD International Conference on Management of Data*, pages 143–154, 2005.
7. P. Bohannon, W. Fan, F. Geerts, X. Jia, and A. Kementsietsidis. Conditional functional dependencies for data cleaning. In *Proceedings of the 23rd International Conference on Data Engineering, ICDE 2007, April 15-20, 2007, The Marmara Hotel, Istanbul, Turkey*, pages 746–755, 2007.
8. C. Borgelt and R. Kruse. Induction of association rules: Apriori implementation. In *Proc. 15th Conf. on Computational Statistics (Compstat 2002, Berlin, Germany)*, pages 395–400, Heidelberg, Germany, 2002. Physika Verlag.
9. F. Chiang and R. J. Miller. Discovering data quality rules. *PVLDB*, 1(1):1166–1177, 2008.
10. G. Cong, W. Fan, F. Geerts, X. Jia, and S. Ma. Improving data quality: Consistency and accuracy. *VLDB*, 2007.
11. Deloitte and Oracle. *Getting Started with Master Data Management*. Deloitte and Oracle White paper, 2005.
12. T. Diallo, N. Novelli, and J.-M. Petit. Discovering (frequent) constant conditional functional dependencies. *International Journal of Data Mining, Modelling and Management (IJDMMM)*, Special issue "Interesting Knowledge Mining":1–20, 2012.
13. J. Euzenat and P. Shvaiko. *Ontology matching*. Springer-Verlag, Heidelberg (DE), 2007.
14. W. Fan, F. Geerts, X. Jia, and A. Kementsietsidis. Conditional functional dependencies for capturing data inconsistencies. *ACM Trans. Database Syst.*, 33(2), 2008.
15. W. Fan, F. Geerts, J. Li, and M. Xiong. Discovering conditional functional dependencies. *IEEE Trans. Knowl. Data Eng.*, 23(5):683–698, 2011.
16. W. Fan, J. Li, S. Ma, N. Tang, and W. Yu. Towards certain fixes with editing rules and master data. In *Proceedings of VLDB'10*, Sept 2010.
17. L. Golab, H. Karloff, F. Korn, D. Srivastava, and B. Yu. On generating near-optimal tableaux for conditional functional dependencies. *Proc. VLDB Endow.*, 1(1):376–390, 2008.
18. Y. Huhtala, J. Karkkainen, P. Porkka, and H. Toivonen. Tane: An efficient algorithm for discovering functional and approximate dependencies. *The Computer Journal*, 42(3):100–111, 1999.
19. J. Kang and J. F. Naughton. On schema matching with opaque column names and data values. In *In SIGMOD*, pages 205–216. ACM Press, 2003.

20. S. Lopes, J.-M. Petit, and L. Lakhal. Efficient discovery of functional dependencies and armstrong relations. In *EDBT*, volume 1777 of *LNCS*, pages 350–364, Konstanz, Germany, 2000. Springer.
21. S. Lopes, J.-M. Petit, and F. Toumani. Discovering interesting inclusion dependencies: application to logical database tuning. *Inf. Syst.*, 27(1):1–19, 2002.
22. D. Loshin. *Master Data Management*. Morgan Kaufmann, 2009.
23. M. J. Maher and D. Srivastava. Chasing constrained tuple-generating dependencies. In *PODS*, pages 128–138, 1996.
24. R. Medina and L. Nourine. Conditional functional dependencies: An fca point of view. In *ICFCA*, pages 161–176, 2010.
25. N. Novelli and R. Cicchetti. Fun: An efficient algorithm for mining functional and embedded dependencies. In *ICDT*, pages 189–203, 2001.
26. N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Discovering frequent closed itemsets for association rules. In *ICDT*, pages 398–416, 1999.
27. D. Power. *A Real Multidomain MDM or a Wannabe*. Orchestra Networks white paper, 2010.
28. E. Rahm and P. A. Bernstein. A survey of approaches to automatic schema matching. *The VLDB Journal*, 10:334–350, December 2001.
29. P. Russom. *Defining Master Data Management*. the data warehouse institute, 2008.
30. C. Wyss, C. Giannella, and E. Robertson. Fastfds: A heuristic-driven, depth-first algorithm for mining functional dependencies from relation instances extended abstract. *Data Warehousing and Knowledge Discovery*, pages 101–110, 2001.
31. M. Zhang, M. Hadjieleftheriou, B. C. Ooi, C. M. Procopiuc, and D. Srivastava. On multi-column foreign key discovery. *PVLDB*, 3(1):805–814, 2010.