Reconstructing Repairs of Census Data Forms as Database Repairs

Leopoldo Bertossi[1], Enrico Franconi[2], and Andrei Lopatenko[2,3]

[1] Carleton University, School of Computer Science, Ottawa, Canada,
bertossi@scs.carleton.ca
[2] Free University of Bozen–Bolzano, Faculty of Computer Science, Italy,
franconi@inf.unibz.it, alopatenko@unibz.it
[3] University of Manchester, Department of Computer Science, UK

Abstract. In the context of consistent query answering (CQA) from inconsistent databases, the notion of repair is fundamental. A repair is a new database instance that minimally differs from the original, inconsistent database, but does satisfy the integrity constraints. Minimality usually refers to a minimal set of tuples on which the two instances differ. In this paper we reexamine the process of correcting census data forms, where the notion of minimal number of changes is natural. Underlying assumptions are made explicit, and on the basis of this, corrections of census questionnaires are characterised as database repairs as introduced in the context of CQA. Minimal number of changes on census questionnaires are represented as database repairs. Other interesting issues addressed here, that are also relevant in the context of database repairs, are the notions of hard and soft constraints for the repair or correction process, and new specific, but natural ways of minimising changes. Finally, on the basis of the database representation, an answer set programming approach to census questionnaires corrections is presented.

1 Introduction

The main goal of this paper is to characterise the problem of amending incorrect census questionnaires—a problem first formalised in [10]—as a special database repair problem, so that a comparison with existing database repair semantics could be carried out.

Given a relational database schema, $\mathcal{S}$, that includes a fixed infinite database domain D, we can consider the first-order language, $\mathcal{L}(\mathcal{S})$, constructed using the predicates in $\mathcal{S}$. Integrity constraints can be expressed as sentences in $\mathcal{L}(\mathcal{S})$. A database instance r can be seen as a set of ground atoms of $\mathcal{L}(\mathcal{S})$, or, alternatively, as a first-order Herbrand structure compatible $\mathcal{S}$. Given a fixed finite set, IC, of integrity constraints, a relational database instance is consistent if it satisfies IC, i.e. $r \models IC$. Otherwise, we say that r is inconsistent. We assume that the set IC is logically consistent in the sense that there is a database instance that satisfies IC. There may be built-in predicates in addition to those in $\mathcal{S}$, but they have the same, fixed extension in every instance.

Database repairs have been introduced in the last few years in the context of consistent query answering (CQA); they are used as an auxiliary notion, for defining the notion of consistent answer to a query in an inconsistent database [1], i.e. a database that does not satisfy a given set IC of integrity constraints. Intuitively speaking, an answer $\bar{t}$ to a query Q posed to a possibly inconsistent database r is consistent if $\bar{t}$ can be obtained as usual answer to Q from every possible *repair*, i.e. a consistent database instance r' that *minimally differs* from r. In other words, consistent answers are invariant under minimal ways of restoring consistency.

One may consider different notions of minimality. The most established notion is the one introduced in [1], where a repair of r is defined as a consistent instance r' such that the symmetric difference $r \Delta r'$ is minimal under set inclusion among the consistent instances that have the same schema (and domain) as r.

Example 1. Consider the database instance $r = \{P(a, b), Q(c, b)\}$ and $IC : \forall x \forall y (P(x, y) \rightarrow Q(x, y))$. Instance r is inconsistent, and the possible repairs are $r_1 = \{Q(c, b)\}, r_2 = \{P(a, b), Q(a, b), Q(c, b)\}$, but not $r_3 = \{P(a, b), Q(a, b)\}$, because $r \Delta r_3 = \{Q(a, b), Q(c, b)\}$ is not minimal (it properly includes $r \Delta r_2 = \{Q(a, b)\}$). □

In [10, 13] an alternative notion of repair was considered, based on a different notion of minimality of difference wrt the original instance. Basically that notion minimises the changes in attribute values. As an example, notice that r_3 in Example 1 could be obtained by changing a for c in Q, i.e. by means of an update affecting one attribute of the tuple in Q. In general, under this notion of repair introduced in [1], consistent instances obtained by updates are not minimal, since they have to be simulated in terms of changes affecting entire tuples, through a deletion followed by an insertion. E.g. under the *value based* notion of repair [13], r_3 in Example 1 would be a repair.

In [10], the notion of repair is applied in the context of census questionnaire forms, where the interest is concentrated on the actual repairs, i.e. on repairing the forms before, say statistical processing, and not on consistent query answering as in [1] (see [5] for a survey of consistent query answering and additional references). As certain fields in the data forms (the records) are corrected, value based repairs are a natural alternative to consider.

As in [10], we can see each questionnaire form as a relational database instance that conforms to the schema

$$Que(Pid, Rel, Sex, Age, Mstat). \tag{1}$$

Here $Att = \{Pid, Rel, Sex, Age, Mstat\}$ is the set of attributes of the schema. Each form contains information about the people living in a same household in reference to a *reference person*, who appears in the first tuple, in the sense that the surrogate attribute *Pid* takes the value 1 on that person. There should be one tuple per person living in the household. We assume that each attribute has a fixed finite domain: $D_{Pid} = \{1, 2, 3, \ldots, N\}$, $D_{Rel} = \{reference, spouse, son, parent,$

$\ldots, null\}$, $D_{Sex} = \{F, M, null\}$, $D_{Age} = \{0, 1, \ldots, 120, null\}$, $D_{Mstat} = \{married, single, widow, null\}$.

In a census questionnaire form, there may be mistakes, not all the values make sense; and before any statistical analysis is performed, the forms that contain mistakes are corrected. The notion of repair and the repair process itself considered in [10] made use of some implicit assumptions. In this paper we want to make those assumptions explicit. The treatment of census questionnaire forms can also be applied in the context of usual relational database instances.

The paper is organised as follows. First, the original problem of census questionnaires repair is fully formalised in Section 2. Then, in Sections 3 and 4 the problem is reduced to a problem of database repair under some special semantics. Finally, an answer set program is sketched in Section 5.

2 Formalising Questionnaires Repairs

It becomes necessary to specify what is a correctly filled questionnaire form. This can be expressed by means of certain *soft integrity constraints IC*,[1] that we know may be violated by a form, but after the correction process, they should be satisfied. There may be other *hard integrity constraints*, *HIC*, that we do no accept to be violated, and if that happens, the corresponding form can be completely discarded. In consequence, only those form that satisfy the hard constraints can be subject to corrections if they do not satisfy the soft constraints. Usually, as in [10], the hard ICs are implicit.

We have the (in [10] implicit) hard constraints, *HIC*, expressing that *Pid* is a primary key, that never takes a *null* value; that always contain the value 1 in *Pid* associated to *reference*; and the latter only to the former:

$$HIC = \{\ \forall x, \bar{x}_1, \bar{x}_2(Que(x, \bar{x}_1) \wedge Que(x, \bar{x}_2) \rightarrow \bar{x}_1 = \bar{x}_2), \forall \bar{x}\ \neg Que(null, \bar{x}),$$
$$\exists \bar{x} Que(1, reference, \bar{x}), \forall x, \bar{x}(Que(x, reference, \bar{x}) \rightarrow x = 1)\ \}. \quad (2)$$

The set *IC* of soft constrains includes the statement that none of the attributes can be *null*, plus other statements that basically express conditions on the relationships between attributes, e.g. it is not possible for a person to be both married and younger than 16 (see [10] for several examples). Actually, all the ICs in *IC* in [10] can be expressed as *denial constraints*, i.e. as formulas of the form

$$\overline{\forall} \neg(Que(\bar{T}_1) \wedge \ldots \wedge Que(\bar{T}_n) \wedge \varphi(\bar{T}_{n+1})), \quad (3)$$

where $\overline{\forall}$ denotes the universal closure; each $\bar{T}_i$ is a tuple of variables and constants; and φ is a formula containing built-in predicates only, say a conjunction

[1] In the literature on consistent query answering, always the ICs have been considered soft, for this reason we keep *IC* as a denotation for the soft ICs. It is to *IC* that we will apply the notions of consistency, repair, and consistent query answer.

of atoms of the form $=, \neq, <, etc.$ Using a common logic programming notation, we simply write instead of (3)

$$:- \ Que(\bar{T}_1), \ldots, Que(\bar{T}_n), \varphi(\bar{T}_{n+1}). \tag{4}$$

Example 2. (based on [10]) Some elements in IC might be: (a) Attributes are not null: $:- \ Que(v, w, x, y, z), x = null$, etc. (b) A spouse of a reference person should be married: $:- \ Que(w, spouse, x, y, z), z \neq married$. (c) Any married person should be at least 16 years old: $:- \ Que(w, x, y, z, married), z < 16$. (d) The age difference between a parent and a child should be at least 15 years. □

In the context of CQA, tuple-based repairs of denial constraints have been investigated in [7]. As observed there, repairs in this sense can be all obtained by tuple deletions only. However, as we commented before, this kind of repair does not capture the repair process of census questionnaire forms (imputation process is the technical term [10]), because that would amount to deleting persons with all their attributes from the form. Instead, in this application, changing values of attributes is the way of restoring consistency wrt IC, making sure that HIC, the number of entries in the form, and the key values are kept. The new values are taken from the attribute domains. Not any change is admissible. Actually, the number of changes in attributes must be minimal.

Example 3. A questionnaire form Que containing the tuple $\bar{t} = Que(2, spouse, F, 30, single)$ (the person number two of the family is declared to be the spouse of the reference person, but she/he is also declared to be single, thus violating a constraint) may have two (alternative) repairs: one $Que'(2, spouse, F, 30, married)$ (where she/he is declared married), the other $Que''(2, daughter, F, 30, single)$ (where she/he is declared to be the daughter of the reference person and still single). □

Definition 1. Given a form r based on $\mathcal{R}$ that satisfies HIC, a *repair* of r is a form r' also based on $\mathcal{R}$, such that:

1. $r' \models HIC$.
2. $r' \models IC$.
3. r' has the same Pid-values as r.
   This establishes a one-to-one correspondence ρ from tuples in r to those in r'. We denote by t' or $\rho(t)$ the tuple in r' corresponding to the tuple t in r.
4. $\Sigma_{\bar{t} \in r} \Sigma_{A \in Att} \delta(\bar{t}.A, \rho(\bar{t}).A)$ is minimal,[2] where the delta function δ gives the value 1 if the arguments are the same, and 0 otherwise. □

Next in [10], *preferred repairs* are defined as those repairs that maximise the number of certain rules that are satisfied. In Example 3, a preferred repair could be $Que'(2, spouse, F, 30, married)$ since there may be a preference rule stating that it is more normal to have a spouse in a family (possibly without daughters) than to have a family with a daughter but without spouse. In this paper, we will concentrate on repairs only.

[2] As usual in databases, $R.A, \bar{t}.A, etc.$ denote the attribute A in relation R, the value of attribute A in tuple $\bar{t}$, resp., etc.

3 Census Forms as Expanded DB Instances

In this Section we will show how a census form can be seen as an instance of a specific DB, which contains both the original form and the amended one.

As in [10], we can decompose relation *Que* into five binary relations that basically correspond to their non key attributes: $A_2(\cdot,\cdot), A_3(\cdot,\cdot), A_4(\cdot,\cdot), A_5(\cdot,\cdot)$, the domain for the first argument in each of these relations is always D_{Pid}. In this way we pass to a one relation schema to a multi-relation schema.

Example 4. The tuple $\bar{t} = Que(2, spouse, F, 30, single)$ in a questionnaire form *Que* is represented by the set of tuples $r(\bar{t}) := \{A_2(2, spouse), A_3(2, F), A_4(2, 30), A_5(2, single)\}$. The form *Que* corresponds then to the database instance $r := \bigcup_{\bar{t} \in Que} r(\bar{t})$. Notice that a form containing the tuple $\bar{t}$ (or the set of tuples $r(\bar{t})$ in the alternative representation) is violating the IC (b) in Example 2. □

Certain approaches to CQA have characterised database repairs as the models of a logical specification [5]. In [2, 11, 10], e.g., new copies of the original predicates have been introduced to denote the repaired predicates and compute their extensions. The old predicates—and their extensions—are forgotten once the necessary data is obtained from them. In the current scenario, since there is a tight relationship between an original form and its corrected version, we need to keep both the old and the new versions. In our case, we represent the repair of a form by means of new predicates $A'_2, \ldots, A'_5$. Initially, they will have the same contents as the A_i. That means, each original form r becomes an instance of the expanded schema $A_2, \ldots, A_5, A'_2, \ldots, A'_5$, and in r, $A_i = A'_i$ for every i.

Example 5. (example 4 continued) The tuple $\bar{t}$ is now represented by the relation

$$r(\bar{t}) = \{\ A_2(2, spouse), A_3(2, F), A_4(2, 30), A_5(2, single),\\ A'_2(2, spouse), A'_3(2, F), A'_4(2, 30), A'_5(2, single)\ \}.$$ □

The repair process will be related to the primed predicates, but keeping the reference to the old form. A repair of a form in this multi-relational representation will have the same extensions as the original form for all its non primed predicates A_i; only the primed predicates are subject to change.

Let us denote by $d\mathcal{R}$ the schema obtained from $\mathcal{R}$ by the decomposition of *Que* into the A_is. $\overline{d\mathcal{R}}$ denotes the schema $d\mathcal{R}$ extended with the A'_is. In consequence, tuples are relations over schema $\overline{d\mathcal{R}}$ (see Example 4). We will keep denoting a form in this representation as a database instance r over the schema $\overline{d\mathcal{R}}$.

4 Census Forms Repairs as DB Repairs

A repair of a form r as introduced in the preceding section can be identified with a mapping ρ that associates with every relation $r(\bar{t})$, a new relation $\rho(r(\bar{t}))$ that keeps the non primed tuples the same, but possibly changes the primed tuples.

Example 6. (example 5 continued) The "tuple" $r(\bar{t})$ in a form r, could have in a repair of r the following image:

$$\rho(r(\bar{t})) = \{\ A_2(2, \mathit{spouse}), A_3(2, F), A_4(2, 30), A_5(2, \mathit{single}),$$
$$A'_2(2, \mathit{spouse}), A'_3(2, F), A'_4(2, 30), A'_5(2, \mathit{married})\}.$$

These should be, in a repair, the only tuples related to 2 (a condition still to be imposed in the formal definition of repair in this framework, see below). Another repair will necessarily have the same non primed tuples, but could have, e.g., the following primed tuples: $A'_2(2,\ \mathit{daughter}),\ A'_3(2, F), A'_4(2, 12), A'_5(2,\ \mathit{single})$; this repair respecting the IC (d) in Example 2. □

In order to impose the conditions on repairs in relation to the correspondence between non primed and primed tuples, we need to introduce the *cardinality constraints* that will ensure that Pids are kept under repairs

$$CC := \{\forall x \forall z \exists y (A_i(x, z) \rightarrow A'_i(x, y)) \mid i = 2, 3, 4, 5\}. \tag{5}$$

Now, HIC will denote the formulas in (2), but with *Que* replaced by the A_i's. For example, we obtain, among others, $\forall x, y, z(A_3(x, y) \wedge A_3(x, z) \rightarrow y = z) \in \mathit{HIC}$. Furthermore, HIC', IC' denote the formulas HIC, IC with the *Que* predicate replaced by the A'_i predicates. For example, $\forall x, y, z(A'_3(x, y) \wedge A'_3(x, z) \rightarrow y = z)$ belongs to HIC'; and

$$:- A'_2(v, w), A'_3(v, x), A'_4(v, y), A'_5(v, z), w = \mathit{spouse}, z \neq \mathit{married}, \tag{6}$$

corresponding to the IC in Example 2(b), belongs to IC'. Now, $\overline{r}|d\mathcal{R}$ denotes the restriction of $\overline{r}$ to schema $d\mathcal{R}$, etc. Notice that the only ICs that mix prime and non primed predicates are those in CC. Finally, we need to compare the primed and non primed parts of an instance over schema $\overline{r}|d\mathcal{R}$. In consequence, for such an instance r, we denote $\Delta(r) := \{A'_i(a, b) \mid \text{exists } d \text{ such that } (a, d) \in A_i \text{ and } d \neq b\}$.

Definition 2. (a) Let r be a form instance over $\overline{d\mathcal{R}}$ such that $r|d\mathcal{R}$ satisfies HIC. A *sub-repair* of r is an instance r^* over the same schema, such that:

1. $r^*|d\mathcal{R} = r|d\mathcal{R}$.
2. $r^* \models CC$.
3. $r^* \models \mathit{HIC}'$.
4. $r^* \models IC'$.

(b) r^* is a *subset-oriented repair* (s.o. repair) of r if it is a sub-repair of r such that $\Delta(r^*)$ is minimal under set inclusion over all instances that are repairs of r.
(c) r^* is a *cardinality-oriented repair* (c.o. repair) of r if it is a repair of r such that the cardinality $|\Delta(r^*)|$ is minimal over all form instances that are repairs of r. □

Proposition 1. Every c.o. repair of r is also a s.o. repair of r.

Proof. Let r^* be a c.o. repair of r. Assume that it is not a s.o. repair. Then, there exists a repair r^{**} of r, such that $\Delta(r^{**}) \subsetneqq \Delta(r^*)$. It follows that $|\Delta(r^{**})| < |\Delta(r^*)|$; a contradiction. □

Notice that this form of repair extends the notion introduced in [1] in the sense that in the minimisation process some predicates—in this case the A_i's—remain fixed, while their primed versions are subject to changes. Another interesting issue, that emerges from keeping in the same instance the old and the repaired part, with ICs like (5) that make the two part interact, is that we no longer repair by simply deleting tuples, but we have to replace them by new ones.[3]. In this way we capture the "correction based repairs" found in [10] into repairs as introduced in [1] or cardinality based repairs (but still tuple oriented) as the approach in [8], that was re-introduced in the context of CQA in [2].

Example 7. The form $r = \{A'_2(1, reference), A'_3(1, M), A'_4(1, 35), A'_5(1, married), A'_2(2, spouse), A'_3(2, F), A'_4(2, 30), A'_5(2, single)\ \}$ is inconsistent if we have the constraint (b) in Example 2.[4] Candidates to repairs of it are:

(a) $r_1 = \{A'_2(1, reference), A'_3(1, M), A'_4(1, 35), A'_5(1, married),\ A'_2(2, spouse),$
$A'_3(2, F),\ A'_4(2, 30),\ A'_5(2, \underline{married})\}$, with $\Delta(r_1) = \{A'_5(2, married)\}$.
(b) $r_2 = \{A'_2(1, reference), A'_3(1, M), A'_4(1, 35), A'_5(1, married),\ A'_2(2, \underline{sister}),$
$A'_3(2, F), A'_4(2, 30),\ A'_5(2, single)\}$, with $\Delta(r_2) = \{A'_2(2, sister)\}$.
(c) $r_3 = \{A'_2(1, reference), A'_3(1, M), A'_4(1, 35), A'_5(1, married),\ A'_2(2, \underline{daughter}),$
$A'_3(2, F),\ A'_4(2, \underline{12}),\ A'_5(2, single)\}$,
with $\Delta(r_3) = \{A'_2(2, daughter),\ A'_4(2, 12)\}$.

Here changes appear underlined. Repairs r_1 and r_2 are c.o. repairs, but not r_3. However the three of them are s.o. repairs. □

According to Section 2, repairs of questionnaire forms should be c.o. repairs. In order to capture c.o. repairs as s.o. repairs, and only those, we need to modify our notion of repair. So far, we have been playing with changes of tuples in relations, however, considering the denial form (4) for the ICs IC, that in IC' take a form like in (6), we may decide to see the built-ins as database tuples, and as such, subject to changes.

More precisely, each census form r can be represented in the form as a set r^{eq} of atoms: $A'_2(1, Y_1^2), eq(Y_1^2, c_{21}), A'_3(1, Y_1^3), eq(Y_1^3, c_{31}),\ \ldots\ , A'_2(2, Y_2^2), eq(Y_2^2, c_{22}),\ \ldots$, where the c_{ij} are constants, possibly *null*, in the corresponding domain, and $eq(\cdot, \cdot)$ is the equality predicate. There might other built-ins as well. On the other side, the ICs in IC' are of the form

$$:- A'_2(X, Y^2),\ A'_3(X, Y^3),\ A'_4(X, Y^4),\ A'_5(X, Y^5),\ eq(Y^2, c_2), \ldots \qquad (7)$$

[3] With denials constraints alone, repairs, as mentioned before, could be all obtained through tuple deletions only. However this is no longer true in the presence of a "referential IC like" constraint like (5).

[4] We are omitting the non primed part, that contains the same values as the primed part.

Example 8. (example 7 continued) The form instance can be written as

$$r^{eq} = \{A'_2(1, Y_1^2), A'_3(1, Y_1^3), A'_4(1, Y_1^4), A'_5(1, Y_1^5), eq(Y_1^2, reference), \ldots,$$
$$A'_2(2, Y_2^2), A'_3(2, Y_2^3), A'_4(2, Y_2^4), A'_5(2, Y_2^5), eq(Y_2^2, spouse), eq(Y_2^4, 30),$$
$$eq(Y_2^5, single), \ldots\};$$

and, for example, (6) can be written as the two denials

$$:- A'_2(v, w), A'_3(v, x), A'_4(v, y), A'_5(v, z), eq(w, spouse), eq(z, single),$$
$$:- A'_2(v, w), A'_3(v, x), A'_4(v, y), A'_5(v, z), eq(w, spouse), eq(z, widow). \quad \square$$

Since the cardinality constrains prevent us from deleting the A'_i atoms, the form can be repaired by deleting equality atoms of the form $eq(X, c)$, what causes the introduction of the non-equality atom $neq(X, c)$.[5]

Example 9. (example 8 continued) Deleting equality tuples from the second line in the form in the example according to the second (and violated) denial there, we obtain, among possibly others, the following repairs:

$$r_1^{eq*} = \{\ldots, A'_2(2, Y_2^2), A'_3(2, Y_2^3), A'_4(2, Y_2^4), A'_5(2, Y_2^5), eq(Y_2^2, spouse), eq(Y_2^4, 30),$$
$$neq(Y_2^5, single), \ldots\};$$

$$r_2^{eq*} = \{\ldots, A'_2(2, Y_2^2), A'_3(2, Y_2^3), A'_4(2, Y_2^4), A'_5(2, Y_2^5), eq(Y_2^4, 30), eq(Y_2^5, single),$$
$$neq(Y_2^5, spouse), \ldots\};$$

$$r_3^{eq*} = \{\ldots, A'_2(2, Y_2^2), A'_3(2, Y_2^3), A'_4(2, Y_2^4), A'_5(2, Y_2^5), neq(Y_2^2, spouse),$$
$$neq(Y_2^4, 30), eq(Y_2^5, single), \ldots\}.$$

In order to determine which of these are s.o. repairs (as defined in Section 1 for database instances) we need to compute the following set differences:

(a) $r^{eq} \Delta r_1^{eq*} = \{eq(Y_2^5, single), neq(Y_2^5, single)\}$.
(b) $r^{eq} \Delta r_2^{eq*} = \{eq(Y_2^2, spouse), neq(Y_2^2, spouse)\}$.
(c) $r^{eq} \Delta r_3^{eq*} = \{eq(Y_2^4, 30), neq(Y_2^4, 30), eq(Y_2^5, single), neq(Y_2^5, single)\}$.

The first two set differences are incomparable, whereas $r^{eq} \Delta r_1^{eq*} \subsetneqq r^{eq} \Delta r_3^{eq*}$. In consequence, we discard r_3^{eq*}.[6] $\square$

We can see each of the sub-repairs in Example 7 as an element of the "class of sub-repairs" determined by $r_1^{eq*}, r_2^{eq*}, r_3^{eq*}$, resp. Notice that any of the actual sub-repairs obtained by admissible instantiations of these classes of extended s.o. repairs correspond to a c.o. repair of the form. The following theorem gives a precise characterisation of c.o. repairs with respect to s.o. repairs.

Theorem 1. For every census questionnaire form r as described in Section 3 that satisfies *HIC*, and a c.o. repair r^* of it according to Definition 2, there is a unique s.o. repair r^{eq*} of r^{eq}, such that r^* can be obtained from r^{eq*} by

[5] We will assume for the moment that we have equality atoms in the denials, but it should not be difficult to extend the treatment to other built-ins.

[6] Notice that we could work everywhere with the one directional set difference $r^{eq*} \setminus r^{eq}$.

instantiating all the variables in r^{eq*} by (non null) values in the corresponding domains. Furthermore, any ground instantiation of the variables in a s.o. repair r^{eq*} that satisfies $HIC' \cup IC'$ is a c.o. repair of the form r.[7] □

5 Answer Set Programming for Form Repairs

It is possible to obtain the forms repairs as the answer sets of a logic program. Notice that in [10] the programs used to compute the c.o. repairs are based on the assumption that at most two changes suffice to repair the forms. The reason is that when more changes are possible, the programs become much more involved. In the context of database repairs, general programs are introduced in [11] (see also [2]), but for ICs with more than two database literals (read, with more than two changes possible), the rules need to consider an exponential number of combinations of literals. A simpler solution to the problem of characterising database repairs as answer sets of logic programs, that keeps the number of rules linear in the number of database literals in the ICs, is given in [3, 4]. There, the programs use annotations, i.e. new constants used as values for a new attribute that expands each of the database tables.

An atom in (outside) the original database is annotated with $\mathbf{t_d}$ ($\mathbf{f_d}$). Annotations $\mathbf{t_a}$ and $\mathbf{f_a}$ are considered *advisory* values, to solve conflicts between the database and the ICs. If an atom gets the derived annotation $\mathbf{f_a}$, it means an advise to make it false, i.e. to delete it from the database. Similarly, an atom that gets the annotation $\mathbf{t_a}$ must be inserted into the database. The logic program should have the effect of repairing the database. Single, local repair steps are obtained by deriving the annotations $\mathbf{t_a}$ or $\mathbf{f_a}$. This is done when each IC is considered in isolation, but there may be interacting ICs, and the repair process may take several steps and should stabilise at some point. In order to achieve this, we use annotations $\mathbf{t}^\star$, $\mathbf{f}^\star$. The latter, for example, groups together the annotations $\mathbf{f_d}$ and $\mathbf{f_a}$ for the same atom. These derived annotations are used to give a feedback to the bodies of the rules that produce the local, single repair steps, so that a propagation of changes is triggered. The annotations $\mathbf{t}^{\star\star}$ and $\mathbf{f}^{\star\star}$ are just used to read off the literals that are inside (resp. outside) a repair.

Next we give an annotated example program. The repair should be read off from the double starred A_i'.

Example 10. (example 5 continued) Consider a set of integrity constraints consisting of constraints a) and b) from example 4 and ICs: c) the partner of the reference person is not married to any other person, d) the spouse of the referenced person has a different sex.

1. Database content.

[7] Notice that by construction of the forms r (and r^{eq}) and their repairs r^{eq*}, the cardinality constraints are automatically satisfied. Notice also that due to the ICs, the variables cannot be instantiated in null values.

(a) A set of facts for non primed predicates given by decomposition of the original database.
$A_2(X,Y) \leftarrow Que(X,Y,\_,\ ,\_)$.
$A_3(X,Y) \leftarrow Que(X,\ ,Y,\_,\_)$.
$A_4(X,Y) \leftarrow Que(X,\_,\_,Y,\_)$.
$A_5(X,Y) \leftarrow Que(X,\_,\_,\_,Y)$.
$domd(X) \leftarrow Que(X,\_,\_,\_,\_)$. ... $domd(X) \leftarrow Que(\_,\_,\_,\_,X)$.

(b) A set of possible values for each attribute.
$EQ_2(X,Y) \leftarrow A_1(X,Y), Y == spouse$.
$EQ_2(X,Y) \leftarrow A_1(X,Y), Y == parent$.
and so on for each attribute and a domain value for it.

(c) Initial values of unrepaired primed predicates.
$A'_2(X,Y,\mathbf{t_d}) \leftarrow A_2(X,Y)$.
...
$A'_5(X,Y,\mathbf{t_d}) \leftarrow A_5(X,Y)$.

2. Rules for closed world assumption.
$A'_2(X,Y,\mathbf{f}^*) \leftarrow notA'_2(X,Y,\mathbf{t_d})$.
$A'_3(X,Y,\mathbf{f}^*) \leftarrow notA'_3(X,Y,\mathbf{t_d})$.
$A'_4(X,Y,\mathbf{f}^*) \leftarrow notA'_4(X,Y,\mathbf{t_d})$.
$A'_5(X,Y,\mathbf{f}^*) \leftarrow notA'_5(X,Y,\mathbf{t_d})$.

3. Annotation rules.
$A'_2(X,Y,\mathbf{f}^*) \leftarrow A'_2(X,Y,\mathbf{f_a})$.
$A'_3(X,Y,\mathbf{f}^*) \leftarrow A'_3(X,Y,\mathbf{f_a})$.
$A'_5(X,Y,\mathbf{f}^*) \leftarrow A'_5(X,Y,\mathbf{f_a})$.
$A'_2(X,Y,\mathbf{t}^*) \leftarrow A'_2(X,Y,\mathbf{t_a})$.
$A'_2(X,Y,\mathbf{t}^*) \leftarrow A'_2(X,Y,\mathbf{t_a})$.
$A'_2(X,Y,\mathbf{t}^*) \leftarrow A'_2(X,Y,\mathbf{t_a})$.
$A'_2(X,Y,\mathbf{t}^*) \leftarrow A'_2(X,Y,\mathbf{t_d})$.
$A'_2(X,Y,\mathbf{t}^*) \leftarrow A'_2(X,Y,\mathbf{t_d})$.
$A'_2(X,Y,\mathbf{t}^*) \leftarrow A'_2(X,Y,\mathbf{t_d})$.

4. Interpretation rules.
$A'_2(X,Y,\mathbf{t}^{**}) \leftarrow A'_2(X,Y,\mathbf{t_a})$.
$A'_2(X,Y,\mathbf{t}^{**}) \leftarrow A'_2(X,Y,\mathbf{t_d}), notA'_2(X,Y,\mathbf{f_a})$.
$A'_2(X,Y,\mathbf{f}^{**}) \leftarrow A'_2(X,Y,\mathbf{f_a})$.
$A'_2(X,Y,\mathbf{f}^{**}) \leftarrow notA'_2(X,Y,\mathbf{t_a}), notA'_2(X,Y,\mathbf{t_d})$.
...
$A'_5(X,Y,\mathbf{t}^{**}) \leftarrow A'_5(X,Y,\mathbf{t_a})$.
$A'_5(X,Y,\mathbf{t}^{**}) \leftarrow A'_5(X,Y,\mathbf{t_d}), notA'_5(X,Y,\mathbf{f_a})$.
$A'_5(X,Y,\mathbf{f}^{**}) \leftarrow A'_5(X,Y,\mathbf{f_a})$.
$A'_5(X,Y,\mathbf{f}^{**}) \leftarrow notA'_5(X,Y,\mathbf{t_a}), notA'_5(X,Y,\mathbf{t_d})$.

5. Rules for integrity constraints

(a) Census integrity constraints.
A spouse of a reference person is married.
$A'_2(X,Y,\mathbf{f_a}) \vee A'_5(X,Z,\mathbf{t_a}) \leftarrow$
$A'_2(X,Y,\mathbf{t}^*) \wedge A'_5(X,Z,\mathbf{f}^*), Y == spouse, Z == married$.
A partner of a reference person is not married.

$A'_2(X, Y, \mathbf{f_a}) \vee A'_5(X, Z, \mathbf{f_a}) \leftarrow$
$A'_2(X, Y, \mathbf{t}^*) \wedge A'_5(X, Z, \mathbf{t}^*) \wedge Y == partner \wedge Z == married.$
A spouse of a reference person has a different sex.
$A'_3(1, S, \mathbf{f_a}) \vee A'_3(X, S_1, \mathbf{f_a}) \vee A'_2(X, Y, \mathbf{f_a}) \leftarrow$
$A'_3(1, S, ts) \wedge A'_3(X, S_1, ts) \wedge A'_2(X, Y, ts) \wedge S == S_1 \wedge Y == spouse$

(b) Referential integrity constraints.
Primed predicates are contained in possible value predicates (but possible value predicates can not be changed in a repair).
$A'_2(X, Y, \mathbf{f_a}) \leftarrow A'_2(X, Y, \mathbf{t}^*), notEQ_2(X, Y)$
$\cdots$
$A'_5(X, Y, \mathbf{f_a}) \leftarrow A'_5(X, Y, \mathbf{t}^*), notEQ_5(X, Y)$
The first-attribute projection of non primed predicates is contained in first-attribute projection of primed predicates (non-primed predicates can not be repaired).
$aux_2(X) \leftarrow A'_2(X, Y, \mathbf{t_a}).$
$aux_2(X) \leftarrow A'_2(X, Y, \mathbf{t_d}), notA'_2(X, Y, \mathbf{f_a})$
$A'_2(X, Y_c, \mathbf{t_a}) \leftarrow$
$A_2(X, Y), notaux_2(X), notA'_2(X, Y_1, \mathbf{t_d}), EQ_2(X, Y_c), choice((X), (Y_c)).$
$\cdots$
$aux_5(X) \leftarrow A'_5(X, Y, \mathbf{t_a}).$
$aux_5(X) \leftarrow A'_5(X, Y, \mathbf{t_d}), notA'_5(X, Y, \mathbf{f_a})$
$A'_5(X, Y_c, \mathbf{t_a}) \leftarrow$
$A_5(X, Y), notaux_5(X), notA'_5(X, Y_1, \mathbf{t_d}), EQ_5(X, Y_c), choice((X), (Y_c)).$

6. Denial constraints for coherence.
$\leftarrow A'_2(X, Y, \mathbf{t_a}), A'_2(X, Y, \mathbf{f_a}).$
$\cdots$
$\leftarrow A'_5(X, Y, \mathbf{t_a}), A'_5(X, Y, \mathbf{f_a}).$ □

6 Conclusions

In this paper, starting from [10], we have formalised census data repairs in precise terms, capturing both implicit and explicit assumptions on census data forms. Those assumptions we represented and classified as hard or soft constraints. The former are assumed to be satisfied both before and after the form correction process. There should be a correspondence of values between the old and the new version of the form. This was captured by means of the cardinality constraints. We called them this way because them, in combination with explicit key constraints, impose a one to one correspondence between the key values in the two versions of the form.

We also managed to capture census form repairs as database repairs in the sense of the definition introduced in [1]. The former minimise the number of values corrected in fields in the form, whereas the latter minimise, wrt set inclusion, the set of tuples changed in the form seen as a database instance. This simple form of reduction could be applied in other scenarios, where the notion of repairs is cardinality based.

Here we just sketch an alternative way of obtaining such a reduction of census form repairs to set inclusion based database repairs: we expand the schema with one predicate $C(\cdot)$, with domain $\mathbb{N}$. This predicate and the domain will allow us to count (in a very limited range; we do not need to go beyond the number of non field keys in the original form, in this case four). Expand an original questionnaire form $\overline{r}$ as in Definition 2 with $C(0)$. Intuitively, there is 0 difference between the A_i and the A'_i. Now, consider the following finite number of ICs:

$$\begin{aligned}
\#0\colon\quad & C(0) \leftarrow C(1) \vee \bigwedge_{i=2}^{5} \forall xy\ (A_i(x,y) \leftrightarrow A'_i(x,y));\\
\#1\colon\quad & C(1) \leftarrow C(2) \vee \exists^{=1} xy(A_2(x,y) \wedge \neg A'_2(x,y)) \wedge\\
& \qquad \bigwedge_{i=3}^{5} \forall xy(A_i(x,y) \leftrightarrow A'_i(x,y)) \vee \cdots;\\
\#2\colon\quad & C(2) \leftarrow C(3) \vee \text{ there are exactly 2 differences;}\\
& \cdots\\
\#4\colon\quad & \cdots.
\end{aligned}$$

Each repair r^* will have an extension $C_{r^*} = \{0, 1, ..., n\}$ ($n = 4$ in the examples). We can compare repairs wrt to set inclusion of the predicate C. The minimal wrt set inclusion will be the same as those minimal wrt number of changes.

According to the representation we chose here for forms and their repairs, where both the old and new values are represented, the kind of minimisation we are performing keeps certain tables fixed (the A_i and the A'_i) and others change (like *eq* and *neq*). It is possible to show that the circumscriptive approach to database repairs presented in [6] can be easily extended to handle this kind of priorities (or minimality with fixed predicates).

We believe that it is not very difficult to extend our approach to handle built-ins other that equality and inequality. This is subject to further work.

Acknowledgements: L. Bertossi was supported by NSERC (grant 250279-02), and also appreciates the support and hospitality received from Enrico Franconi when visiting the University of Bolzano.

References

1. Arenas, M.; Bertossi, L.; and Chomicki, J. Consistent Query Answers in Inconsistent Databases. In *Proc. ACM Symposium on Principles of Database Systems (ACM PODS'99, Philadelphia)*, ACM Press, 1999, pp. 68–79.
2. Arenas, M.; Bertossi, L.; and Chomicki, J. Answer Sets for Consistent Query Answers. Theory and Practice of Logic Programming, 3, 4&5, 2003, pp. 393-424.
3. Barcelo, P. and Bertossi, L. Logic Programs for Querying Inconsistent Databases. Proc. Fifth International Symposium on Practical Aspects of Declarative Languages (PADL 2003). Springer Lecture Notes in Computer Science 2562, 2003, pp. 208–222.
4. Barcelo, P., Bertossi and Bravo, L. Characterizing and Computing Semantically Correct Answers from Databases with Annotated Logic and Answer Sets. In 'Semantics in Databases', Springer LNCS 2582, 2003, pp. 1–27.

5. Bertossi, L.; and Chomicki, J. Query Answering in Inconsistent Databases. In 'Logics for Emerging Applications of Databases', J. Chomicki, R. van der Meyden, and G. Saake (eds.), Springer, 2003.
6. Bertossi, L., Schwind, C. "Database Repairs and Analytic Tableaux". To appear in Annals of Mathematics and Artificial Intelligence journal.
7. Chomicki, J.; and Marcinkowski, J. Minimal-Change Integrity Maintenance Using Tuple Deletions. arXiv.org paper cs.DB/0212004.
8. Dalal, M. "Investigations into a Theory of Knowledge Base Revision". In Proc. AAAI 1988.
9. Eiter, T.; Faber, W.; Leone, N.; Pfeifer, G. Declarative Problem-Solving in DLV. In *Logic-Based Artificial Intelligence*, J. Minker (ed.), Kluwer, 2000, pp. 79–103.
10. Franconi, E.; Laureti Palma, A.; Leone, N.; Perri, S.; and Scarcello, F. Census Data Repair: a Challenging Application of Disjunctive Logic Programming. In Logic for Programming, Artificial Intelligence, and Reasoning (LPAR 2001). Springer LNCS 2250, 2001, pp. 561-578.
11. Greco, G.; Greco, S.; and Zumpano, E. A Logic Programming Approach to the Integration, Repairing and Querying of Inconsistent Databases. In Proc. 17th International Conference on Logic Programming, ICLP'01, Ph. Codognet (ed.), LNCS 2237, Springer, 2001, pp. 348–364.
12. Wang, H. and Zaniolo, C. Nonmonotonic reasoning in $\mathcal{LDL}^{++}$. In Jack Minker, editor, *Logic-Based Artificial Intelligence*, pages 523–544. Kluwer Academic Publishers, Dordrecht, 2000.
13. Wijsen, J. Condensed Representation of Database Repairs for Consistent Query Answering. In Database Theory - ICDT 2003, Springer LNCS 2572 ,2003, pp. 378-393.