Census Data Repair: a challenging application of Disjunctive Logic Programming

Enrico Franconi[1], Antonio Laureti Palma[2],
Nicola Leone[3], Simona Perri[3], and Francesco Scarcello[4]

[1] Dept. of Computer Science, Univ. of Manchester, UK
franconi@cs.man.ac.uk
[2] ISTAT, National Statistical Institute, Roma, Italy
lauretip@istat.it
[3] Dept. of Mathematics, Univ. of Calabria, Rende (CS), Italy
leone@unical.it, sperri@si.deis.unical.it
[4] DEIS, Univ. of Calabria, Rende (CS), Italy
scarcello@deis.unical.it

Abstract. Census data provide valuable insights on the economic, social and demographic conditions and trends occurring in a country. Census data is collected by means of millions of questionnaires, each one including the details of the persons living together in the same house. Before the data from the questionnaires is sent to the statisticians to be analysed, a cleaning phase (called "imputation") is performed, in order to eliminate consistency problems, missing answers, or errors. It is important that the imputation step is done without altering the statistical validity of the collected data. The contribution of this paper is twofold. On the one hand, it provides a clear and well-founded declarative semantics to questionnaires and to the imputation problem. On the other hand, a correct modular encoding of the problem in the disjunctive logic programming language DLP^w, supported by the **DLV** system, is shown. It turns out that DLP^w is very well-suited for this goal. Census data repair appears to be a challenging application area for disjunctive logic programming.

1 Introduction

In most countries, a census of population is hold every five or ten years. The census consistently updates the most fundamental source of information about a country. The collected information provides a statistical portrait of the country and its people. The census is the only reliable source of detailed data for small population groups such as the very elderly or specific industrial and occupational categories, and for small geographic areas such as, for example, a city neighbourhood. Census data can be used in many ways such as providing the boundaries for federal electoral districts, helping businesses to select new manufacturing sites, analysing markets for products and services, and to develop employment policies and programs.

Each family compiles a questionnaire, which includes the details of the head of the household (householder) together with the other persons living in the house. The forms are collected and then analysed by statistical agencies in order to modify and update the population household statistics. Questionnaires may be incomplete and/or may contain inconsistent information. It turns out that a significant proportion of the forms have problems of this kind, which if not corrected could severely alter the updated statistics. There are millions of questionnaires to be processed each time, and it is desirable to employ methodologies to detect (*edit*) and automatically resolve (*impute*) errors in the forms. All the methodologies currently employed by the census agencies – including the most popular Fellegi and Holt methodology [7] – are based on statistical principles, which do not have a clear semantics with respect to the complexity of the various errors that may happen, so that their behaviour while imputing the data is often unpredictable. Moreover, it is difficult to compare the various methodologies, since they are defined only on a procedural basis.

The first contribution of this paper is to define in a pure classical logic setting the semantics of the edit and imputation problems. The approach is very general and very expressive, since it makes use of full First Order Logic. In Section 4, the framework is defined in its generality, while making use of a simplified basic questionnaire as a running example. The advantage of this approach is the ability to specify in a declarative fashion the characteristics of the census questionnaire, the intra- and inter-questions constraints, and the additional statistical information required to guide the corrections. This is the first proposal in the literature making use of Logic at all the stages of the definition of the problem.

The second important contribution of the paper is to provide a correct encoding of the edit and imputation problems in an executable specification using an extension of disjunctive logic programming. In Section 6, we provide the rationale of the modular encoding, together with partial translations of the running example. For this purpose, disjunctive logic programming extended with two kinds of constraints has been used (see Section 5). The idea is that the preferred models of the logic program encoding the problem correspond to the repairs of a given questionnaire. It turns out that the solution of this challenging problem requires most of the expressive capabilities of disjunctive logic programming with constraints. Importantly, this disjunctive logic programming language is supported by **DLV** [5, 6], a system which is freely available on the web, and can therefore be used to obtain an immediate implementation. Preliminary results of an experimentation activity carried out at the Italian National Statistical Institute show that the approach is viable also from the viewpoint of performance.

2 The Census Questionnaire

A *household* is a group of persons who occupy the whole or part of a housing unit and provide themselves with food and possibly other essentials for living. It is customary that, during the census of population periodically carried out by every country, a single questionnaire is associated to each household, involving questions about each member of the household. The notion of household may vary from country to country and it is in general to be distinguished from the notion of family. In the context of this paper we will not consider all the subtleties which make each census different from each other, but we will maintain an abstract level which could be adapted to the different situations, and nonetheless it will be in agreement with the recommendations for the 2000 European censuses of population [9]. In this Section, a basic questionnaire will be introduced, including all the core attributes (also called *variables*) considered in most censuses; the semantic definitions and the methodology presented in this paper can be easily extended to handle all the variables of a complete questionnaire.

Each questionnaire is centred around the notion of *reference person*. For statistical purposes, all persons living in private households are identified from their relationship to the reference member of the household. It is outside the scope of this paper to understand how the various administrations define and select the one reference person in a household to whom all other persons in the household report or designate their relationship. In order to identify members of the household, an integer number is assigned to each person, with the number "1" reserved to identify the reference person. The following classification of persons living in a private household by `relationship` to the household's reference person is adopted in a basic questionnaire: Spouse; Partner; Child; Parent; Other relative of reference person, spouse or partner; Non-relative of reference person, spouse, or partner.

The other *structural* attributes – to be specified for each member of the household – considered to be part of a basic questionnaire are: `Sex`: Male, Female; `Age`: integer number; `Marital Status`: Single, Married, Divorced, Widowed.

A census form to be compiled by the members of a household will then contain, for each member, questions for at least each of the above attributes.

3 Dealing with Missing and Inconsistent Data

When the millions of compiled forms are returned to the national central statistical agency to be analysed, they may contain many errors, such as *missing* or *inconsistent data*. For example, it is possible that a person declared to be the spouse of the reference person, but at the same time he/she forgot to declare a marital status, or he/she declared to be single, or to be 6 years old. Before the data from the questionnaires is sent to the statisticians to be analysed, a cleaning phase is performed, in order to eliminate missing values or inconsistencies. It is very important that this step is done *without altering the (statistical) validity* of the collected data. For example, if the spouse declared to be 6 years old, and if there are other arguments to *enforce* that he/she is actually the spouse of the reference person (for example, he/she may have additionally declared to be married), then his/her age should be changed to make the data consistent: any age, say, greater than 16 years would be fine if only absolute validity of the data is taken into account. However, the age should be changed to an age in agreement with the average age for a spouse in his/her conditions, and not with a value which may alter the statistics. In this case, it would be more sensible to change the age to, say, 36 years, rather than to 96 years (there are very few people of that age, and this would alter the statistics of old people) or to 16 years (there are very few people who are married at that age). Other corrections can be done *deterministically* – in the sense that there is only one valid change, which is also a statistically valid change – for example for a spouse who forgot to specify the marital status, and for which asserting that he/she is married does not introduce any other inconsistency.

For these reasons, a collection of *edit* rules is introduced by the statistical agencies for each questionnaire class, in order to understand whether a questionnaire is consistent, and to guide the repair of inconsistent questionnaires. The process of automatically producing a set of statistically consistent questionnaires from the raw ones is called *imputation*.

4 Formalisation of the Imputation Problem

We define a questionnaire $\mathcal{Q}$ as a pair $\langle r, E\rangle$, where r is a relation (i.e., a finite set of tuples) over the schema

$$sch : \texttt{R}(\underline{\texttt{PersonId}}, \texttt{Relationship}, \texttt{Sex}, \texttt{Age}, \texttt{MaritalStatus})$$

that encodes the answers of a household to the questionnaire, and E is a finite set of first order (short: FO) formulas encoding the edit rules that r should satisfy. Note that, in this paper, we will only use relations with the above schema *sch*. Thus, hereafter, we will always omit to specify the schema of a relation. Hence, e.g., by "all relations" we will mean "all relations over schema *sch*". Note also that the imputation problem is defined with respect single questionnaires only. Therefore, the edit rules are associated to each single questionnaire. Of course, the edit rules are the same for all questionnaires.

The `PersonId` attribute is key for any relation r over *sch*, and the domain of each attribute is the finite set of values specified in Section 2, with in addition the special *null* value. Null values encode missing answers in the questionnaire. The edit rules, encoded through the FO sentences E, play the role of integrity constraints over the relation R. In these formulas the predicate R, additional comparison predicates over the domains, and constants for domain values may occur. We say that $\mathcal{Q}$ is a *complete questionnaire* if no null value occur in the tuples of r, and that Q is a *consistent questionnaire* if it is complete and r is model for the theory E, i.e., $r \models E$.

Example 1 Consider a questionnaire $\mathcal{Q}_1 = \langle r_1, E_1\rangle$, where r_1 is the following relation:

PersonId	Relationship	Sex	Age	Marital Status
1	reference	F	31	married
2	spouse	M	39	married

and where E_1 is the following set of edit rules:

1. a spouse of the reference person should be married (but not necessarily vice versa):
$\forall x.\ \exists y\ v_1\ v_2.\ R(x, \text{spouse}, v_1, v_2, y) \rightarrow y = \text{married};$
2. (formal) cohabitant partners of the reference person can not be married with somebody else (i.e., they should be either single, or divorced, or widowed):
$\forall x.\ \exists y\ v_1\ v_2.\ R(x, \text{partner}, v_1, v_2, y) \rightarrow y \neq \text{married};$
3. the reference person cannot be married with more than one person:
$\forall x\ y.\ \exists v_1\ v_2\ v_3\ v_4\ v_5\ v_6.\ R(x, spouse, v_1, v_2, v_3) \wedge R(y, spouse, v_4, v_5, v_6) \rightarrow x = y$
4. the reference person and its spouse should have a different sex:
$\forall x.\ \exists y\ v_1\ v_2\ v_3\ z\ v_4\ v_5.\ R(x, spouse, y, v_1, v_2) \wedge R(1, v_3, z, v_4, v_5) \rightarrow y \neq z;$
5. any married person should be at least 16 years old:
$\forall x.\ \exists y\ v_1\ v_2.\ R(x, v_1, v_2, y, \text{married}) \rightarrow y \geq 16;$
6. the difference in age between a parent and a child should be bigger than 12 years:
$\forall x.\ \exists y\ z\ v_1\ v_2\ v_3\ v_4\ v_5.\ R(x, \text{child}, v_1, y, v_2) \rightarrow (R(1, v_3, v_4, z, v_5) \wedge (z - y) \geq 12),$
and
$\forall x.\ \exists y\ z\ v_1\ v_2\ v_3\ v_4\ v_5.\ R(x, \text{parent}, v_1, y, v_2) \rightarrow (R(1, v_3, v_4, z, v_5) \wedge (y - z) \geq 12)$

The questionnaire $\mathcal{Q}_1$ is consistent and complete, as no null value occur in r_1 and all the edit rules are satisfied by r_1.

If a questionnaire $\mathcal{Q} = \langle r, E \rangle$ is not complete and/or not consistent, then we should repair it, providing some new corrected relation denoting a complete and consistent questionnaire.

Definition 1 (Admissible Repair). Let $\mathcal{Q} = \langle r, E \rangle$ be a questionnaire and ρ a mapping from tuples over *sch* to tuples over *sch*. We say that such a mapping is a *repair* for $\mathcal{Q}$. For a relation $\bar{r}$, denote by $\rho(\bar{r})$ the relation $\{\rho(t) \mid t \in \bar{r}\}$. ρ is an *admissible repair* for $\mathcal{Q}$ if $\langle \rho(r), E \rangle$ is a complete and consistent questionnaire.

Example 2 Consider the questionnaire $Q_2 = \langle r_2, E_1 \rangle$, where E_1 is the set of edit rules in Example 1, and r_2 is the following relation:

PersonId	Relationship	Sex	Age	Marital Status
1	reference	F	31	married
2	spouse	M	39	null

An admissible repair for this questionnaire is the mapping ρ_2 saying that the reference person has an unmarried partner, i.e., the mapping that changes the second tuple to $\langle 2, \text{partner}, M, 39, \text{single} \rangle$ and leaves untouched the first tuple.

Starting from the observation that the errors in the compilation of a form are exceptions, we make our best guess for a corrected questionnaire by minimising the changes to the original relation leading to a complete and consistent questionnaire. For this purpose, we introduce a measure of the number of changes that a repair makes to the relation of a questionnaire. For two tuples $t = \langle v_1, \ldots, v_m \rangle$ and $t' = \langle v'_1, \ldots, v'_m \rangle$, let $dist(t, t')$ be the number of values v_j occurring in t that differ from the corresponding values in t', i.e., such that $v_j \neq v'_j$. Let $\mathcal{Q} = \langle r, E \rangle$ be a questionnaire and ρ a repair for it; we define $changes(\rho, \mathcal{Q}) = \sum_{t \in r} dist(t, \rho(t))$.

Definition 2 (Minimal Repair). An admissible repair ρ for a questionnaire $\mathcal{Q}$ is *minimal* if $changes(\rho, \mathcal{Q})$ is the minimum number of changes over all possible admissible repairs for $\mathcal{Q}$.

Example 3 Consider again the questionnaire $Q_2 = \langle r_2, E_1 \rangle$ in Example 2. The questionnaire has only a minimal repair, ρ'_2, namely the one where the null value is replaced by the 'married' marital status for the spouse of the reference person:

PersonId	Relationship	Sex	Age	Marital Status
1	reference	F	31	married
2	spouse	M	39	married

Note that the repair ρ_2 in the previous example is not minimal. Indeed, it changes both the relationship and the marital status for person 2 and hence $changes(\rho_2, \mathcal{Q}_2) = 2$, while $changes(\rho'_2, \mathcal{Q}_2) = 1$.

In general, there are many possible minimal repairs for an inconsistent and/or incomplete questionnaire. We should then choose one of the minimal repairs without altering the statistical properties of the census. This can be obtained by exploiting some extra information about the census domain, possibly old statistics concerning the same population. Such information can be encoded through a set of FO formulas P that we call *preference rules*, used for expressing some preferences over the repaired relation r', defined as $r' = \rho(r)$, given r. To distinguish the two relations within the same formula, we use a different predicate symbol R' for the repaired relation r': therefore, formulas in P refer to both the predicates R and R'. Intuitively, a preference rule specifies a preferred way to repair the data of a person under some circumstances. The preference rules should be satisfied by as many persons in the household as possible. In order to introduce such a measure, we exploit that each person is uniquely identified by her/his PersonId, and we define preference rules as open formulas ϕ_x, with one free variable x ranging over the domain of the key attribute PersonId. Therefore, these rules are in fact FO queries, telling how many tuples (i.e., how many persons) do actually satisfy them.

Given a substitution θ for the variable x in the domain of the key attribute, $\phi_{\theta x}$ denotes the FO sentence obtained by replacing the variable x according to θ. For a preference rule ϕ_x and the relations r and r', we define

$$\phi_x(r, r') \;=\; \{\, \theta x \mid [r, r'] \models \phi_{\theta x} \}.$$

where $[r, r']$ denotes the interpretation induced by the two relation instances r and r' over predicates R and R', respectively. Intuitively, the relation $\phi_x(r, r')$ contains the members of the household whose repair r' satisfies the preference rule ϕ. If $|\phi_x(r, r')|$ denotes the number of tuples in $\phi_x(r, r')$, for a set of preference rules P we define $|P(r, r')| = \sum_{\phi \in P} \phi(r, r')$.

Example 4 Consider the following preference rule ϕ_x:

$$\begin{aligned} \forall s_1, a_1, s_2, a_2.\, & R(1, \text{reference}, s_1, a_1, \text{married}) \wedge R(x, \mathsf{null}, s_2, a_2, \text{married}) \rightarrow \\ & R'(x, \text{spouse}, s_2, a_2, \text{married}) \end{aligned}$$

This rule expresses that it is likely for a married person living in the household, whose relationship with the reference person is unknown, to be his/her spouse.

Preference rules are used to establish an ordering within the minimal repairs of a questionnaire. Intuitively, when we change some value in order to repair inconsistencies and/or to replace null values, we want to satisfy as many preference rules as possible.

Now we are ready to give a formal definition of the imputation problem.

Definition 3 (Preferred Repair). A *preferred repair* ρ for a questionnaire $\mathcal{Q}$ is a minimal repair such that the number of satisfied preference rules $|P(r, \rho(r))|$ is the greatest over all minimal repairs for $\mathcal{Q}$.

Definition 4 (Imputation Problem). Let $\mathcal{Q} = \langle r, E \rangle$ be a questionnaire and P a set of preference rules, expressed as open FO formulas over predicates R and R'. The *imputation problem* for $\mathcal{Q}$ with respect to P is the problem of finding a preferred repair ρ for $\mathcal{Q}$.

Please note that each imputation problem involves only the data of a single household – i.e., it should be computed for each census form at a time without considered the rest of the data. Thus, the size of the problem is relatively small. This includes the number of family members, the number of the attributes defining a questionnaire and the size of their domains, and the number of the edit and preference rules which are in total usually less between 50 and 100 for a standard complete census questionnaire. A crucial parameter is the size of the domains; as we have seen, these sizes are not usually large, with the exception of the domain of the age attribute. To overcome this problem, usually the domain of the age attribute is partitioned into age intervals. According to our experience, edit rules are also not a source of complexity: they tend to have a rather standard structure.

Example 5 Consider the questionnaire $Q_3 = \langle r_3, E_1 \rangle$, where E_1 is the set of edit rules in Example 1, and r_3 is the following relation:

PersonId	Relationship	Sex	Age	Marital Status
1	reference	F	31	married
2	null	M	39	married

Note that there is no unique way to provide the missing 'relationship' value for the second tuple of r_3. The person identified by 2 may be another relative or a non-relative belonging to the household, or the spouse of the reference person (but not a partner, child, or parent). All these alternative belong to the set of the minimal repairs for Q_3. However, as specified by the preference rule stated above, most married couples live together, and hence we should probably prefer the "spouse" repair. Let ρ_3 be the minimal repair that leaves untouched the first tuple of r_3 and maps its second tuple to $\langle 2, \text{spouse}, M, 39, \text{married} \rangle$. Moreover, let $r_3' = \rho_3(r_3)$. Then, $\phi_x(r_3, r_3') = \{1, 2\}$, because ϕ_x is satisfied by $[r_3, r_3']$ both substituting x with 1 and substituting x with 2. Consider a repair ρ_3' where the second tuple is mapped to $\langle 2, \text{relative}, M, 39, \text{married} \rangle$. In this case, $\phi_x(r_3, \rho_3'(r_3)) = \{1\}$, because the tuple for person 2 does not satisfy ϕ_x.

Thus, as expected, the unique solution of the imputation problem for Q_3 w.r.t. P is the repair ρ_3 giving the following relation $r_3' = \rho_3(r_3)$.

PersonId	Relationship	Sex	Age	Marital Status
1	reference	F	31	married
2	spouse	M	39	married

Example 6 Consider the questionnaire $Q_4 = \langle r_4, E_1 \rangle$, where E_1 is the set of edit rules in Example 1, and r_4 is the following relation:

PersonId	Relationship	Sex	Age	Marital Status
1	reference	F	31	married
2	parent	M	6	widowed

It is easy to see that the set of minimal repairs of Q_4 includes any questionnaire which is obtained by Q_4 by substituting the age of the parent with any value between 47 and 120 (the maximum age). We should not prefer a too big or a too small age difference for average parent-child pairs. This can be expressed through the following preference rule ϕ'_x, stating that we prefer age differences in the range $(25, 35)$:

$$\forall s'_1, s_2, s'_2, a'_1, a_2, a'_2, m'_1, m_2, m'_2.$$
$$R(x, \text{parent}, s_2, a_2, m_2) \ \wedge \ R'(1, \text{reference}, s'_1, a'_1, m'_1) \ \rightarrow$$
$$R'(x, \text{parent}, s'_2, a'_2, m'_2) \ \wedge \ 25 \leq (a'_2 - a'_1) \leq 35$$

The following is not among the minimal repairs of Q_4, since it does not involve a minimal number of changes:

PersonId	Relationship	Sex	Age	Marital Status
1	reference	F	31	married
2	child	M	6	single

5 Disjunctive Logic Programming with Constraints

In this section we describe disjunctive logic programming language with constraints (DLP^w) [4], which is supported by the **DLV** system [5, 6]. We will use DLP^w in Section 6 to implement our approach to repair inconsistent census data. We first provide an informal description of the DLP^w language by examples, and we then supply a formal definition of the syntax and semantics of DLP^w.

5.1 DLP^w by Examples

Consider the problem SCHEDULING, consisting in the scheduling of course examinations. We want to assign course exams to time slots in such a way that no two exams are assigned to the same time slot if the corresponding courses have some student in common – we call such courses "incompatible". Supposing that there are three time slots available, ts_1, ts_2 and ts_3, we express the problem in DLP^w by the following program $\mathcal{P}_{sch}$:

$$r_1 : \ assign(X, ts_1) \vee assign(X, ts_2) \vee assign(X, ts_3) :\!- \ course(X)$$
$$s_1 : \ \ :\!- \ assign(X, S), assign(Y, S), incompatible(X, Y)$$

Here we assumed that the courses and the pair of incompatible courses are specified by input facts with predicate $course$ and $incompatible$, respectively. Rule r_1 says that each course is assigned to either one of the three time slots ts_1, ts_2 or ts_3; the strong constraint s_1 (a rule with empty head) expresses that no two incompatible courses can be assigned to the same time slot. In general, the presence of strong constraints modifies the semantics of a program by discarding all models which do not satisfy some of them. Clearly, it may happen that no model satisfies all constraints. For instance, in a specific instance of above problem, there could be no way to assign courses to time slots without having some overlapping between incompatible courses. In this case, the problem does not admit any solution. However, in real life, one is often satisfied with an approximate solution, in which constraints are satisfied as much as possible. In this light, the problem at hand can be restated as follows (APPROX SCHEDULING): "assign courses to time slots trying not to overlap incompatible courses". In order to express this problem we introduce the notion of *weak* constraint, as shown by the following program $\mathcal{P}_{a\_sch}$:

$$r_1: \ assign(X, ts_1) \vee assign(X, ts_2) \vee assign(X, ts_3) \text{ :− } course(X)$$
$$w_1: \ \text{:}\sim assign(X, S), assign(Y, S), incompatible(X, Y)$$

From a syntactical point of view, a weak constraint is like a strong one where the implication symbol :− is replaced by :∼ . The semantics of weak constraints minimises the number of violated instances of constraints. An informal reading of the above weak constraint w_1 is: "preferably, do not assign the courses X and Y to the same time slot if they are incompatible". Note that the above two programs $\mathcal{P}_{sch}$ and $\mathcal{P}_{a\_sch}$ have exactly the same preferred models if all incompatible courses can be assigned to different time slots (i.e., if the problem admits an "exact" solution).

In general, the informal meaning of a weak constraint, say, :∼ B, is "try to falsify B" or "B is preferably false", etc. Weak constraints are very powerful for capturing the concept of "preference" in commonsense reasoning.

Since preferences may have, in real life, different priorities, weak constraints in DLP^w can be assigned different priorities as well, according to their "importance"[1]. For example, consider the case when incompatibilities among courses have different defeasibility degree – e.g., basic courses with common students should be considered with a higher degree of incompatibility than advanced courses. Consider the following problem (SCHEDULING WITH PRIORITIES): "schedule courses by trying to avoid overlapping between basic incompatible courses first, and then by trying to avoid overlapping between advanced incompatible courses" (i.e., privilege the elimination of overlapping between basic incompatible courses). If incompatibilities are specified through input facts with predicates $basic\_incompatible$ and $advanced\_incompatible$, we can represent SCHEDULING WITH PRIORITIES by the following program $\mathcal{P}_{p\_sch}$:

$$r_1: \ assign(X, ts_1) \vee assign(X, ts_2) \vee assign(X, ts_3) \text{ :− } course(X)$$
$$w_2: \ \text{:}\sim assign(X, S), assign(Y, S), basic\_incompatible(X, Y) \quad [:2]$$
$$w_3: \ \text{:}\sim assign(X, S), assign(Y, S), advanced\_incompatible(X, Y) \quad [:1]$$

The weak constraint w_2 is defined "stronger than" w_3, since w_2 has a higher priority (2) than w_3 (1). The preferred models (called *best models*) of the above program are the assignments of courses to time slots that minimise the number of overlappings between basic incompatible courses and, among these, the assignments which minimise the number of overlappings between advanced incompatible courses.

5.2 Syntax

A *term* is either a constant or a variable[2]. An *atom* is of the form $\mathtt{a}(\mathtt{t_1}, ..., \mathtt{t_n})$, where $\mathtt{a}$ is a *predicate* of arity *n* and $\mathtt{t_1}, ..., \mathtt{t_n}$ are terms. A *literal* is either a *positive* literal p or a *negative* literal $\mathtt{not}\ p$, where p is an atom.

[1] Note that priorities are meaningless for strong constraints, since all of them *must* be satisfied.
[2] Note that function symbols are not part of datalog.

A *(disjunctive) rule* r is a clause of the form

$$\mathtt{a_1 \vee \cdots \vee a_n :\!-\, b_1, \cdots, b_k, not\ b_{k+1}, \cdots, not\ b_m.} \qquad \mathtt{n \geq 1,\ m \geq 0}$$

where $\mathtt{a_1, \cdots, a_n, b_1, \cdots, b_m}$ are atoms. The disjunction $a_1 \vee \cdots \vee a_n$ is the *head* of r, while the conjunction $b_1, ..., b_k,$ `not` $b_{k+1}, ...,$ `not` b_m is the *body* of r. If $n = 1$ (i.e., the head is $\vee$-free), then r is *normal*; if $m = k$ (the body is `not` -free), then r is *positive*.

A *strong constraint* (integrity constraint) has the form $:\!-\ \mathtt{L_1, \cdots, L_m}$., where each $\mathtt{L_i}$, $1 \leq\ i \leq m$, is a literal; it is a rule with empty head.

A *weak constraint* has the form $:\sim \mathtt{L_1, \cdots, L_m}.[:\mathtt{l}]$, where each L_i, $1 \leq\ i \leq m$, is a literal and l is a term. l denotes the *priority layer*, once the weak constraint has been instantiated l should be a positive integer[3]. If the priority level l is omitted, then it defaults to 1.

The language DLP^w include all possible rules and constraints (weak and strong). A DLP^w program $\mathcal{P}$ is a finite subset of the language DLP^w.

5.3 Semantics

Informally, the semantics of a DLP^w program $\mathcal{P}$ is given by the stable models of the set of the rules of $\mathcal{P}$ satisfying all strong constraints and minimising the number of violated weak constraints according to the prioritisation.

The Herbrand Universe, the Herbrand Base, and the ground instantiation $ground(\mathcal{P})$ of a DLP^w program $\mathcal{P}$ are defined in the usual way. Let P, S, and W be, respectively, the set of ground instances of the rules, the strong constraints, and the weak constraints of a DLP^w program $\mathcal{P}$. An interpretation I for $\mathcal{P}$ is a subset of the Herbrand Base of $\mathcal{P}$ (i.e., it is a set of ground atoms). A positive ground literal A is true w.r.t. I if $A \in I$; otherwise it is false w.r.t. I. A negative ground literal `not` A is true w.r.t. I if $A \notin I$; otherwise it is false w.r.t. I. A ground rule $r \in P$ is satisfied in I if some head literal of r is true or some body literal of r is false w.r.t. I. A constraint (strong or weak) c is satisfied in I, if some literal of c is false w.r.t. I. An interpretation satisfying all rules of P is a model of P. The *Gelfond-Lifschitz transformation* of P with respect to I, denoted by P^I, is the positive program defined as follows:

$$\begin{aligned} P^I = \{\ & a_1 \vee \cdots \vee a_n :\!-\ b_1, \cdots, b_k | \\ & a_1 \vee \cdots \vee a_n :\!-\ b_1, \cdots, b_k, \neg b_{k+1}, \cdots, \neg b_m \in P \ \ and \ \ b_i \notin I,\ k < i \leq m\} \end{aligned}$$

I is a *stable model* for P if I is a subset-minimal model of P^I. A *candidate model* of $\mathcal{P}$ is a stable model of P which satisfies each strong constraint in S.

For the weak constraints, only those candidate models are considered which minimise the number of the violated (weak) constraints in the greatest priority layer, and among them those which minimise the number of the violated constraints in the previous layer, etc. This can be expressed by an objective function for $\mathcal{P}$ and a candidate model M:

$$\begin{aligned} & f(1) = 1 \\ & f(n) = f(n-1) \cdot WC + 1, \qquad \text{if } n > 1 \\ & H_M = \textstyle\sum_{i=1}^{max_l} (f(i) \cdot ||violated(M, i)||) \end{aligned}$$

[3] Note that the language of **DLV** supports also weights besides layers. Since we do not use them in our application, we omit the weights to simplify the presentation.

where WC denotes the total number of weak constraints in $\mathcal{P}$, max_l denotes the maximum priority layer in $\mathcal{P}$, $||violated(M, i)||$ denotes the number of weak constraints in layer i which are violated w.r.t. M. A candidate model of $\mathcal{P}$ for which H_M is minimal among all candidate models of $\mathcal{P}$ is a *best model* of $\mathcal{P}$.

As an example, consider the following program $\mathcal{P}_s$

$$\begin{array}{llll} a \vee b :- c. & \quad\quad :\sim a, c. & [: 2] \\ c. & \quad\quad :\sim b. & [: 1] \end{array}$$

The stable models for the set $\{\ c.\ a \vee b :- c.\}$ of ground rules of this example are $M_1 = \{\mathtt{a}, \mathtt{c}\}$ and $M_2 = \{\mathtt{b}, \mathtt{c}\}$, they are also the candidate models, since there is no strong constraint. In this case, $WC = 2$, and $max_l = 2$. Thus, $f(1) = 1$, $f(2) = 3$, $H_{M_1} = 3$, and $H_{M_2} = 1$. So M_2 is preferred over M_1 (M_2 is a best model of $\mathcal{P}_s$). Indeed, M_1 violates the more important weak constraint having level 2; while M_2 violates only a weak constraint of level 1.

6 Solving the Imputation Problem

In this section we show how to solve the imputation problem using a disjunctive logic program with weak constraints [4] whose best models correspond to the preferred repairs of a given questionnaire. This program first verifies if the input data satisfies the edit rules or contains null values, i.e., if the questionnaire is consistent and complete. If this is the case, then the program has a unique best model corresponding to the given questionnaire, unmodified. Otherwise, the program corrects automatically the data, and each of its best models encodes a preferred repair, i.e., it maximises the number of satisfied preference rules).

6.1 Running Example

As a running example we consider the questionnaire $Q_5 = \langle r_5, E_1 \rangle$ and the preference rules ϕ_x of Example 4 and ϕ'_x of Example 6 . E_1 is the set of edit rules in Example 1, and r_5 is the following relation:

PersonId	Relationship	Sex	Age	Marital Status
1	reference	F	31	single
2	parent	null	12	widowed

The questionnaire is incomplete (it is not specified the sex for the person "2") and inconsistent with respect to the edit rule 6 from E_1 (the person "2" is a parent of the reference person and she/he is 12 years old).

6.2 Data Representation

Each component of a household is identified by an integer number. The number "1" identifies the reference person. The questionnaire relation is encoded by the following

EDB binary predicates: `relationship, age, maritalstatus`, and `sex`. For each atom over these predicates, the first argument corresponds to the identification number of a person, while the domain for the second argument depends on the predicate, as described below:

- `relationship`: reference, spouse, partner, son, parent, otherRelative, nonRelative,
- `sex`: male, female,
- `marital status`: single, married, divorced, widowed,
- `age`: integer number.

For instance, an atom `relationship(2,spouse)` indicates that the household component with identification number "2" is the spouse of the reference person. In our running example the relation r_5 is represented by the following set of facts:

```
{relationship(1,reference), sex(1,female), age(1,31),
 maritalstatus(1,single)
 relationship(2,parent), age(2,12), maritalstatus(2,widowed)}
```

The relation corresponding to a preferred repair for the given questionnaire is encoded through a set of facts with the same name as the predicates encoding the input relation, but capitalised. For instance, `Age(1,33)` means that the reference person is aged 33 in the preferred repair. Therefore, the "output" relation corresponding to a preferred repair is represented by the following IDB predicates: `Relationship, Age, MaritalStatus`, and `Sex`.

6.3 Description of the DLP^w Program

The DLP^w program consists of the following five modules automatically generated from the edit rules:

- detection of errors and missing data;
- generation of possible correction of data values;
- admissible repairs check;
- minimal repairs generation;
- preferred repairs generation.

Next, we describe the five modules. To help the intuition, we carry out an informal description by referring to the running example.

Detection of Errors and Missing Data. This module verifies that the input data is correct (i.e., there is no inconsistency w.r.t some edit rule) and that there is no missing information about the components of the household (i.e., for each person, there exists a tuple for each of the predicates). If either some null value or some error is detected in any input atom, this module derives a corresponding atom over the predicates `wrongRelationship, wrongAge, wrongMaritalStatus`, and `wrongSex`. Missing values are identified by using negation. For instance, consider the input predicate `relationship`. The following rules detect if we miss the relationship information for some member of the household:

```
specifiedRelationship(X) :- relationship(X,Y).
wrongRelationship(X,missing) :- not specifiedRelationship(X),
                                person(X).
```

If a relationship of a person X is not specified, the wrongRelationship(X,missing) atom is derived. This means that it is necessary to complete the data. In the running example, since the sex of the person "2" is not specified , the module derives the atom wrongSex(2,missing).

In order to detect errors, the DLP^w program includes rules corresponding to the edit rules of the questionnaire, that single out inconsistencies. If an edit rule is violated, then the corresponding rule of the module detects this situation and provides some atoms, which may be responsible for this violation. For instance, consider the edit rule 5 in the running example, which says that any married person should be at least 16 years old. The following rule suggests that, in the case of violation for a person X, either the age or the marital status of X is wrong:

```
wrongMaritalStatus(X,married) v wrongAge(X,Z):-
    maritalstatus(X,married),age(X,Z),Z<16.
```

Note that, for presentation clarity, we made a simplifying assumption. We assume that a violated edit rule does not need to change more than one atom to be satisfied (e.g. either marital status or age can be changed in the above rule, but we assume that we do not need to change both). Even if this assumption is plausible we can easily relax it by modifying the encoding.

Generation of Possible Corrections of Data Values. The previous module derives atoms of the form wrongSomething corresponding to wrong values that caused the violation of some edit rule. For each of them, we should provide a new value for the repaired questionnaire. This is accomplished by disjunctive rules whose heads contain an atom for each domain value, representing a candidate change for the guilty atom.

For instance, the following rule guesses a value for the (output) sex predicate Sex, if the sex is wrong:

```
Sex(X,male) v Sex(X,female) :- wrongSex(X,_).
```

In the previous module, since we derived wrongSex(2,missing), a new value for the output predicate Sex is guessed.

Admissible Repairs Check. Once we have guessed a repair for the given questionnaire, we have to check if the repair is admissible. This is accomplished by integrity constraints over the output predicates that encode the edit rules of the questionnaire. Thus, the output atoms occurring in the candidate models of the program so far correspond to admissible repairs for the questionnaire.

For instance, consider the edit rule 1 stating that the spouse of the reference person should be married, and the edit rule 5 stating that any non single person should be at least 16 years old. They are encoded by the following constraints:

```
:- Relationship(X,spouse), MaritalStatus(X,Y), Y<>married.
:- MaritalStatus(X,Y), Y<>single, Age(X,Z), Z<16.
```

A candidate model for the program of the running example is[4]:

```
A={wrongSex(2,missing), wrongAge(2,12), Relationship(1,reference),
   Sex(1,female), Age(1,31), MaritalStatus(1,single),
   Relationship(2,parent), Sex(2,female), Age(2,70),
   MaritalStatus(2,widowed)}
```

corresponding to the following repair:

PersonId	Relationship	Sex	Age	Marital Status
1	reference	F	31	single
2	parent	F	70	widowed

Another candidate model for this program is

```
A'={wrongSex(2,missing), wrongRelationship(2,parent),
    Relationship(1,reference), Sex(1,female), Age(1,31),
    MaritalStatus(1,single) Relationship(2,child), Sex(2,male),
    Age(2,12), MaritalStatus(2,single)}
```

corresponding to the following repair:

PersonId	Relationship	Sex	Age	Marital Status
1	reference	F	31	single
2	child	M	12	single

Minimal Repairs Generation. To enforce the computation of the only candidate models corresponding to minimal repairs of the questionnaire, we exploit the weak constraints extension of disjunctive logic programming. Using weak constraints, we can assign a penalty to each value change we make to the input questionnaire. This is accomplished by the following constraints:

```
:~ relationship(X,Y), Relationship(X,Z), Z<>Y. [:2]
:~ age(X,Y), Age(X,Z), Z<>Y. [:2]
:~ sex(X,Y), Sex(X,Z), Z<>Y. [:2]
:~ maritalstatus(X,Y), MaritalStatus(X,Z), Z<>Y. [:2]
```

Note that we use level 2 priority for the weak constraints above, because minimisation of changes has the highest priority in the imputation problem (as defined in Section 4), and we will use level 1 priority for specifying less important properties below.

The candidate model `A'` above violates three weak constraints, while the candidate model `A` violates two weak constraints. Thus, `A` is a best model of the program, and it corresponds to a minimal repair for Q_5.

[4] We omit to specify the input facts again in the models.

Preferred-Repairs Generation. To maximise the number of satisfied preference rules (Definition 3), we use again weak constraints. The module contains a weak constraint for each preference rule. This constraint assigns a penalty for each violation of the corresponding preference rule. These weak constraints have level 1 priority, since they have a lower priority than the weak constraints enforcing minimisations of changes. Indeed, by definition of the imputation problem, the preferred repairs should be chosen among the minimal repairs for the questionnaire. Thus, the candidate model of the program minimising the cost of weak constraints encodes the preferred repair for the questionnaire.

For instance, the following weak constraint corresponds to the preference rule of the running example stating that, if the reference person is married and some other married person is in the household, then we prefer the situation in which they are married each other:

```
:~ MaritalStatus(1,married), MaritalStatus(X,married),
   Relationship(X,Y), Y <> spouse, X <> 1.    [:1]
```

The candidate model A is a minimal repair but it is not the preferred repair for Q_5. Indeed, it does not satisfy the preference rule stating that the age difference between a parent and a child should be in the range (25,35), on average.

The following candidate model is a best model of the program and corresponds to a preferred repair of the questionnaire.

```
A''={wrongSex(2,missing), wrongAge(2,12), Relationship(1,reference),
     Sex(1,female), Age(1,31), MaritalStatus(1,single),
     Relationship(2,parent), Sex(2,female), Age(2,60),
     MaritalStatus(2,widowed)}
```

Remark 1. To show the feasibility of our approach, we used the executable specification language DLP^w, supported by the **DLV** system. It is worthwhile noting that disjunction is not strictly necessary in our program, and could be replaced by unstratified negation (with a polynomial blow up of the program size). Weak constraints, on the contrary, are strictly needed to implement the semantics of preferred repairs, and cannot be rewritten into normal logic rules. Thus, normal logic programs cannot be used to implement our semantics, while DLP^w provides a natural and elegant support for the implementation.

7 Related Work

There are various methodologies available to statisticians for dealing with the millions of records that have to be cleaned [10]. The most widely used for census data is the Fellegi and Holt methodology (FH [7]). More recently, the New Imputation Methodology (NIM [2]) has been introduced as an improvement over FH.

In both FH and NIM edit rules are written in a declarative fashion, and they are used to check the consistency of a household questionnaire; the consistency check is usually implemented with active rules in a relational database. When a questionnaire fails to satisfy the edit rules, the information needed for the imputation phase is borrowed from a similar questionnaire that passed the edit rules (which is called a *donor* form). NIM initially searches for donors that match the failed record on as many as possible of the involved attribute values. It then analyses the edit rules to determine the *minimum* number of non-matching attributes between the failed record and that particular donor, such that the imputed record will eventually pass the edits. Various subsets of the attributes are imputed to determine which is the optimum imputation for a failed edit household. It is crucial that in the imputed record the joint distribution of attributes is maintained. In contrast to NIM, FH first determines the minimum number of attributes to impute and then performs the imputation, possibly by searching for donors. According to the experiments, changing the order of these operations allows NIM to efficiently solve larger and more complex imputation problems.

Both the above methods do not have a declarative semantics. In particular, various implementations slightly differ in the imputation phase, making it impossible to understand what gets repaired and why. In addition, they are computationally expensive, since both rely inherently on donors, which have to be found in the large database of size of the entire census.

Our proposal has two clear advantages over FH and derivative methods. First of all, it is equipped with formal declarative semantics, which unambiguously specifies the meaning and the behaviour of the imputation task. Surprisingly, there has been no previous attempt in the field to devise explicitly a formal semantics to the imputation task. This semantics is not in contrast with what in principle should be the behaviour of the FH and NIM procedures. However, we have noticed that the various algorithms currently used do produce different results which are hard to interpret and compare given the absence of a declarative semantics. We hope that our clear semantics will help in understanding and possibly fixing those differences. Secondly, we have clearly separated the consistency check phase, the abstract imputation phase, and the preference selection phase. While we give precise and complete definition for the former phases, we have defined the latter preference selection phase up to the choice of a mechanism behind the preference rules. We leave open the definition of them to the statisticians, who may choose their own preferred strategy, such as a donor strategy – with possibly different similarity measures, and with global or local comparisons – or a strategy based on statistical histories, or others.

Our approach is highly modular. In principle, each questionnaire is processed and corrected alone, independently on all other census data. This modularity property allows for a high degree of parallelism (several questionnaires can be processed in parallel on different machines) and ensures a higher computational load. Indeed, even if the correction of a questionnaire may require an exponential time, this time is exponential in the size of a single questionnaire and in the number of edit rules. This is true up to the computation of the preference rules, which – depending on the strategy chosen by the statisticians – may again look at the whole set of questionnaires. The advantage of our proposal is to single out this latter phase and to have a principled way of relating it with the abstract imputation problem. We also propose a provably correct implementation of the abstract imputation problem in DLP^w. We have implemented our approach on the disjunctive logic programming system **DLV**, and we have started an experimentation activity at the Italian Statistical Agency (ISTAT). The results of the experiments show the feasibility of this approach with realistic questionnaires (composed by the edit rules and the form data), if only the abstract imputation problem is taken into account, i.e., if the preference rules are given.

To our knowledge, there has been in the literature only a proposal using a logic-based approach to the edit and the imputation problems [3]. However, this approach does not explicitly define a declarative semantics of the imputation problem, and most importantly it suffers from a combinatorial explosion of the encoding of the problem in propositional logic. Moreover, the main focus of [3] is on the problems of consistency and redundancy of edit rules. The imputation problem is solved – again in a propositional setting – still in an ad-hoc manner by using donors taken from correct questionnaires.

The problem of correcting collection of data with missing or inconsistent values also arises in the context of integration of knowledge from multiple sources, e.g., in data warehouses. Some recent logical approaches to querying and repairing inconsistent databases have been proposed in [1, 8]. Such approaches are based on *subset-minimal* repairs, and it is possible to specify some preference on the possible corrections. The use of subset-minimal repairs is more coarse grained than our approach based on *minimal change* followed by a *preference* criterion. In fact, our notion of minimality is based on the finer grained measure of the number of changes within tuples. For example, let us consider two admissible repairs of a questionnaire, each one having only one tuple changed with respect to the original questionnaire; this is the case of the two admissible repairs for Q_2 proposed in example 2 (ρ_2) and example 3 (ρ'_2). These two repairs are ranked equal according to a subset-minimal criterion, while our minimal change criterion selects only the repair(s) where the least number of attributes was changed, namely ρ'_2. In addition, we further allow general preference rules among the set of minimal repairs, expressed in a declarative way, with the goal of producing a single preferred repair.

We believe that our minimal change criterion could be applied as a basic selection criterion also for the problems in [1, 8], since it leads to more sensible choices. Moreover, even if it is hard to precisely compare the complexity of the approaches pursued by [1, 8] with ours – since the problems are different (computation of a consistent answer in all minimal repairs versus the computation of the minimal repairs) – a general complexity argument applies as follows. First note that the imputation problem is a search problem, and that we refer here to the complexity of the problem to decide whether a value of a tuple gets a certain value in some minimal repair of the questionnaire. In this context, choices among different solutions based on weights or cardinalities of the solutions are typically feasible in a more efficient way, compared with choices based on the subset-minimality criterion. More specifically, in our case, the adoption of a general subset-minimal criterion would raise the complexity from Δ_2^p (our upper bound, given by the fragment we use of DLP^w with a limited – head-cycle-free – disjunction [4]) to Σ_2^p.

8 Conclusions

In this paper we have formally defined the semantics of the edit and imputation problems for census data, and we have provided a correct encoding in disjunctive logic programming with constraints. We believe that our framework is quite general and can be applied to contexts other than census data repair.

We have implemented our approach on the disjunctive logic programming system **DLV**, and we have started an experimentation activity at the Italian Statistical Agency

(ISTAT). Preliminary results of experiments show the feasibility of this approach with realistic data.

We plan to have an extensive experimentation with real census data provided by the Italian Statistical Agency – including their donor-based strategy for computing the preference rules – and to compare the outcome of our approach with the well established imputation methodologies based on statistics. We also want to devise a methodology for edit and preference rule design, and to precisely characterise the computational properties of the abstract imputation problem.

References

1. M. Arenas, L. Bertossi, and J. Chomicki. Specifying and Querying Database repairs using Logic Programs with Exceptions. In *Proceedings International Conference on Flexible Query Answering*, pages 27–41, 2000.
2. Michael Bankier. Experience with the new imputation methodology used in the 1996 canadian censuses with extensions for future censuses. In *Statistical Commission and Economic Commission for Europe, Conference of European statisticians, UN/ECE work session on statistical Data Editing*, Rome, Italy, 1999.
3. Renato Bruni and Antonio Sassano. Errors detection and correction in large scale data collecting. In F. Hoffmann et al., editor, *Proc. of the Fourth International Symposium on Intelligent Data Analysis (IDA-2001)*, volume 2189 of *LNCS*, pages 84–94. Springer-Verlag, 2001.
4. Francesco Buccafurri, Nicola Leone, and Pasquale Rullo. Enhancing Disjunctive Datalog by Constraints. *IEEE Transactions on Knowledge and Data Engineering*, 12(5):845–860, 2000.
5. Thomas Eiter, Nicola Leone, Cristinel Mateis, Gerald Pfeifer, and Francesco Scarcello. The KR System `dlv`: Progress Report, Comparisons and Benchmarks. In Anthony G. Cohn, Lenhart Schubert, and Stuart C. Shapiro, editors, *Proceedings Sixth International Conference on Principles of Knowledge Representation and Reasoning (KR'98)*, pages 406–417. Morgan Kaufmann Publishers, 1998.
6. Wolfgang Faber, Nicola Leone, and Gerald Pfeifer. Experimenting with heuristics for answer set programming. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI) 2001*, Washington, USA, August 2001. To appear.
7. I. P. Fellegi and D. Holt. A systematic approach to automatic edit and imputation. *Journal of the American Statistical Association*, 71(353):17–35, March 1976.
8. G. Greco, S. Greco, and E. Zumpano. A Logic Programming Approach to the Integration, Repairing and Queriyng of Inconsistent Databases. In *Proceedings International Conference on Logic Programming*, 2001.
9. United Nations Economic Commission for Europe and the Statistical Office of the European Communities. Recommendations for the 2000 censuses of population and housing in the ECE region. Technical Report Statistical Standards And Studies – No. 49, UN/ECE Statistical Division, 1998.
10. William E. Winkler. State of statistical data editing and current research problems. In *Statistical Commission and Economic Commission for Europe, Conference of European statisticians, UN/ECE work session on statistical Data Editing*, Rome, Italy, 1999.