A Logical Formalisation of the Fellegi-Holt Method of Data Cleaning

Agnes Boskovitz[1], Rajeev Goré[1], and Markus Hegland[2]

[1] Research School of Information Sciences and Engineering
Australian National University, Canberra ACT 0200 Australia
[2] Mathematical Sciences Institute
Australian National University, Canberra ACT 0200 Australia
{agnes.boskovitz, rajeev.gore, markus.hegland}@anu.edu.au

Abstract. The Fellegi-Holt method automatically "corrects" data that fail some predefined requirements. Computer implementations of the method were used in many national statistics agencies but are less used now because they are slow. We recast the method in propositional logic, and show that many of its results are well-known results in propositional logic. In particular we show that the Fellegi-Holt method of "edit generation" is essentially the same as a technique for automating logical deduction called resolution. Since modern implementations of resolution are capable of handling large problems efficiently, they might lead to more efficient implementations of the Fellegi-Holt method.

1 Introduction

Data errors are everywhere. While minor errors might be inconsequential, data errors can often seriously affect data analysis, and ultimately lead to wrong conclusions. Detecting and correcting errors to clean data is a large part of the time-consuming preprocessing stage of any data-intensive project.

Obvious strategies for dealing with potentially erroneous data, like revisiting the original data sources or doing nothing, have clear disadvantages. If the locations of the errors are known, we can discard dirty data, again with clear disadvantages. More practically, statistical techniques can be used to estimate a likely correction. Fortunately, errors can often be located using the data's internal dependencies, or redundancies, as seen in the following (contrived) example.

Example, part 1. (This will be a running example throughout this paper.) A census of school students includes a data record showing a person who is aged 6, has a driver's licence and is in Grade 8 of school. Domain knowledge gives:

Requirement 1: A person with a driver's licence must be in at least Grade 11.
Requirement 2: A person in Grade 7 or higher must be at least 10 years old.

The given data record fails both of the requirements. Since school grade level appears in both of them, it might seem that the record can be corrected by changing the grade level. However the two requirements cannot be satisfied by

changing only the grade level. The difficulty arises because there is an additional requirement that can be deduced from these, namely:

Requirement 3: A person with a driver's licence must be at least 10 years old.

Requirement 3 does not involve school grade level, but is still failed by the data record. In fact two fields must be changed to satisfy all of the requirements. Of course in reality something much stronger than Requirement 3 holds, but Requirement 3 is the logical deduction from the first two Requirements.

In this paper we address the problem of automating the task of correcting record data when each record must satisfy deterministic requirements. After stating our assumptions in Section 2, we describe in Section 3 the elegant contribution of Fellegi and Holt [1], who recognised the need to find "all" deducible requirements to decide where to correct a record. They developed an algorithm for finding deducible requirements, and proved that it finds enough requirements to determine which fields must be changed to correct an incorrect record. The Fellegi-Holt method was used for many years in government statistical agencies, but is less used now because it is too slow for many practical applications [2].

There have been many published refinements to the Fellegi-Holt method [3–7]. However, they can be confusing, because the method is expressed in terms of records which *fail* requirements, whereas the deducible requirements are about records which *satisfy* requirements. We think a formalisation that helps people to reason about the method itself, especially about the logical deduction of new requirements, should decrease the level of confusion.

Symbolic logic is a well-developed and rigorous framework for formalising reasoning and performing logical deduction. Although the problem of finding "all" deductions is NP-complete, computer implementations for automatically finding logical deductions are being continually improved. In Sections 4 and 5 we formalise the Fellegi-Holt method in logic, giving us two benefits. Firstly we gain a rigorous perspective with which to reason about the problem of data correction and about Fellegi and Holt's solution. Secondly we gain the potential to use automated deduction techniques to speed up the method. In our formalisation, the Fellegi-Holt results become well-known results in classical propositional logic. In particular, Fellegi-Holt deduction is essentially the same as a standard technique called resolution deduction. The vast field of results for resolution deduction translate to Fellegi-Holt deduction, giving a new and potentially faster technique for implementing the Fellegi-Holt method. Section 6 gives the results of some experiments which prove our concept. Although other published formalisations in logic solve the data-correction problem, no other work formalises the Fellegi-Holt method itself in logic.

2 Assumptions and Problem Statement

We assume that we are dealing with data arranged in records. A *record* R is an N-tuple $(R_1, \ldots, R_N)$ where $R_j \in A_j$. Here A_j is the *domain* of the j^{th} *field*,

and we assume that it is a finite set. We assume that N is fixed for all records under consideration. The set of all such N-tuples is called the *domain space* $D = A_1 \times \cdots \times A_N = \prod_1^N A_j$.

Example, part 2. The Example has three fields:

$A_1 = \{5, 6, \ldots, 20\}$ (relevant ages), which we relabel A_{age}.
$A_2 = \{\text{Y}, \text{N}\}$ (whether someone has a driver's licence), which we relabel A_{driver}.
$A_3 = \{1, 2, \ldots, 12\}$ (school grade levels), which we relabel A_{grade}.
The domain space is $D = A_{\text{age}} \times A_{\text{driver}} \times A_{\text{grade}}$.
The record in this example is $R = (6, \text{Y}, 8)$.

We assume that potential errors in the data are specified by deterministic *requirements* obtained from domain knowledge, and that the requirements apply to one record at a time. The requirements given in the Example are typical. In the statistical literature, such requirements are usually formalised as *edits*, where an edit consists of the elements of D which are *incorrect* according to the corresponding requirement. Elements of the edit e are *incorrect* according to e or *fail* e. Elements of $D \setminus e$ are *correct* according to e or *satisfy* e.

Example, part 3. The edits corresponding to Requirements 1-3 are:

$$\begin{array}{rccccc} e_1 = & A_{\text{age}} & \times & \{\text{Y}\} & \times & \{1, 2, \ldots, 10\}, \\ e_2 = & \{5, 6, \ldots, 9\} & \times & A_{\text{driver}} & \times & \{7, 8, \ldots, 12\}, \\ e_3 = & \{5, 6, \ldots, 9\} & \times & \{\text{Y}\} & \times & A_{\text{grade}}\,. \end{array}$$

Since $R \in e_1$, $R \in e_2$ and $R \in e_3$, R is said to fail all three edits.

In our logical formalisation in Section 4, we represent the requirements as logical formulae called *checks*, which represent the *correctness* of a record. To each edit there corresponds a check.

We can now give a formal statement of the data correction problem: given a record R and a set of requirements, find an N-tuple R' which differs from R in as few (weighted) fields as possible and satisfies all of the requirements. Solving the data correction problem involves two steps:

1. *Editing:* the process of testing a data record against edits; and
2. *Imputation:* the correction of incorrect data, which itself takes two steps:
   (a) *error localisation:* finding a smallest (weighted) set of fields which can be altered to correct the record, and then
   (b) changing those fields so as to preserve the original frequency distributions of the data as far as possible (sometimes also called "imputation").

In this paper we concentrate on error localisation, which is the area in which the Fellegi-Holt method is slow.

3 The Fellegi-Holt (FH) Method

Fellegi and Holt's method hinges on two aspects:

1. *FH edit generation:* a systematic process for generating new edits from a given set of edits (described below in Section 3.1); and
2. *FH error localisation* and *guarantee*: a method of finding a smallest set of fields that is guaranteed to yield a correction (described in Section 3.2).

3.1 FH Generation of New Edits

Fellegi and Holt generate edits in terms of *normal edits*, where a normal edit is an edit e of the form $\prod_{j=1}^{N} A_j^e$, with $A_j^e \subseteq A_j$. The edits in the Example, part 3, are normal edits. Any edit can be written as the union of a set of normal edits.

Given a set E of normal edits where each edit e is written $\prod_{j=1}^{N} A_j^e$, and given a field i, the *FH-generated edit* on E with generating field i is $\mathrm{FHG}(i, E)$:

$$\mathrm{FHG}(i,E) = \prod_{j=1}^{i-1} \bigcap_{e\in E} A_j^e \quad \times \quad \bigcup_{e\in E} A_i^e \quad \times \prod_{j=i+1}^{N} \bigcap_{e\in E} A_j^e. \tag{1}$$

Example, part 4. $e_3 = \mathrm{FHG}(\mathrm{grade}, \{e_1, e_2\})$.

The following proposition tells us that any record that satisfies all edits in E must also satisfy $\mathrm{FHG}(i, E)$, so the edit generation process produces an acceptable edit. The result applies because $\mathrm{FHG}(i, E) \subseteq \bigcup_{e\in E} e$.

Proposition 1 (Soundness). *[1, Lemma 1] Let E be a set of normal edits, i be a field, and R be a record such that for all $e \in E, R \notin e$. Then $R \notin \mathrm{FHG}(i, E)$.*

Given a starting set E of edits, the FH edit generation process can be applied to all subsets of E, on all fields. The newly generated edits can then be added to E and the process repeated until no new edits can be generated. The process will eventually terminate because the domain is finite.

The slowness of this process slows down the Fellegi-Holt method. The process can be sped up by excluding any edit which is a subset of (*dominated* by) another generated edit because the FH error localisation guarantee still applies [3]. We call the unique set of edits left at the end of this process the *set of maximal generated edits*, written $\mathrm{MGE}(E)$, or just MGE when the context is clear.

3.2 FH Error Localisation Method and Guarantee

Given an arbitrary set of edits, the FH error localisation method depends on:

1. If a record R fails a normal edit e, then any correction to R must change at least one field j that is *involved* in e, ie $A_j^e \neq A_j$. Otherwise the change in field j cannot affect R's correctness according to e.

2. Hence if R fails some normal edits then any correction of R must change a set of fields C which includes an involved field from each failed edit. Such a set of fields C is called a *covering set*.
3. There is no guarantee that every covering set of the failed set of edits will yield a correction to R. However if the edits are the failed edits in the *MGE*, then *any* covering set will yield a correction to R (Theorem 1 below).
4. The FH error localisation method finds a smallest covering set, giving a solution to the error localisation problem.

Theorem 1 (Error localisation guarantee). *Let R be a record and let E be a set of normal edits such that the edit $D = \prod_{j=1}^{N} A_j \notin \mathrm{MGE}(E)$. Let $E' = \{e \in \mathrm{MGE}(E) \mid R \in e\}$ be those edits in MGE(E) that are failed by R. Then there exists a covering set of E', and for every covering set C there exists an N-tuple R' which (1) is identical to R on fields outside C, and (2) satisfies all the edits in E. ([1, Theorem 1, Corollaries 1, 2] and [3, Lemma 4, Corollaries 1,2])*

Example, part 5. Both edits in $E = \{e_1, e_2\}$ are failed by the given record, but the smallest covering set of E, namely {grade}, does not yield a correction. The record fails every edit in $\mathrm{MGE}(E) = \{e_1, e_2, e_3\}$, so $E' = \{e_1, e_2, e_3\}$. Any pair of fields is a smallest covering set of the MGE and will yield a correction.

3.3 Overcoming the Slowness of Edit Generation

The Fellegi-Holt method is slow because of the slowness of edit generation. Improvements to the edit generation process have been published over the years, eg [3–5], although there is disagreement about whether the "Field Code Forest" algorithm finds the whole MGE [6]. Fellegi and Holt proposed using "essentially new" edit generation, a method that produces fewer edits than the MGE, but which requires much analysis prior to the generation of each new edit.

There has also been work on methods that avoid the large scale edit generation, and instead solve the editing problem record by record. Garfinkel, Liepins and Kunnathur [3–5] and Winkler [7] propose a cutting plane algorithm which avoids finding all implied edits, and instead finds only those that are needed for each record. Barcaroli [8] formalises the problem in first order logic, and then uses a systematic search through possible corrections to find a solution. Bruni and Sassano [9] formalise the problem in propositional logic, use a propositional satisfiability solver for edit validation and error detection, and then use integer programming for error localisation and imputation. Franconi et al [10] use disjunctive logic programming with constraints, solving both the editing and imputation problem. In each case, there is no preliminary edit generation but the workload for each record is increased.

In contrast, the Fellegi-Holt idea is to do a large amount of work even before the data is received, with correspondingly less work for each individual record. The Fellegi-Holt method would therefore be best applied to datasets such as population censuses where there is a long time between the finalisation of the questionnaire and data entry. It is during this time that the large scale edit generation might be completed. Once the data is entered the editing itself should

take less time than the record-by-record approach. Thus our logical formalisation, which builds on the formalisation of Bruni and Sassano, keeps the full generation of the MGE.

4 Translation of the Fellegi-Holt Method to Logic

In this section we represent the FH method in terms of classical propositional logic. We first represent each of the FH constructs as a corresponding logical construct, and then represent the key error localisation result.

As in [9], we represent individual field values by *atoms*, for example p^6_{age} stands for "age = 6". The set of atoms is $\{p^v_j \mid j = 1, \ldots, N, v \in A_j\}$. In the Example we use "age", "driver" and "grade" to represent 1, 2 and 3 respectively.

Formulae are built from the atoms using the propositional connectives $\neg, \vee, \wedge$ and $\rightarrow$. We represent edits as formulae called *checks*.

Example, part 6. The edits e_1, e_2 and e_3 are represented by the checks

$$\begin{aligned}
\gamma_1 &= \neg\,(&& && p^{\text{Y}}_{\text{driver}} && \wedge && (p^1_{\text{grade}} \vee \cdots \vee p^{10}_{\text{grade}})\,) \\
\gamma_2 &= \neg\,(&& (p^5_{\text{age}} \vee \cdots \vee p^9_{\text{age}}) && \wedge && && (p^7_{\text{grade}} \vee \cdots \vee p^{12}_{\text{grade}})\,) \\
\gamma_3 &= \neg\,(&& (p^5_{\text{age}} \vee \cdots \vee p^9_{\text{age}}) && \wedge && p^{\text{Y}}_{\text{driver}} && &&)\,.
\end{aligned}$$

Each N-tuple R is represented by a *truth function* $f_R : \text{Atoms} \rightarrow \{\text{true}, \text{false}\}$ which can be extended in the usual way to truth functions on formulae. In any N-tuple, each component takes exactly one value, so we require every truth function to map to "true" the following sets of checks, called *axioms*:

Axiom 1: $\neg p^v_j \vee \neg p^w_j$, for all $j = 1, \ldots, N$ and for $v \neq w$. (Each field of an N-tuple can take at most one value.)

Axiom 2: $\bigvee_{v \in A_j} p^v_j$, for all $j = 1, \ldots, N$. (Each field of an N-tuple must take at least one value.)

Example, part 7. $R = (6, \text{Y}, 8)$, so $f_R(p^6_{\text{age}}) = f_R(p^{\text{Y}}_{\text{driver}}) = f_R(p^8_{\text{grade}}) =$ true. For all other atoms p, $f_R(p) =$ false. For the checks, $f_R(\gamma_1) = f_R(\gamma_2) = f_R(\gamma_3) =$ false.

If $f_R(\gamma) =$ false then the record R is said to *fail* the check γ or to be *incorrect* according to γ. In the Example, R fails γ_1, γ_2 and γ_3. If $f_R(\gamma) =$ true then the record R is said to *satisfy* the check γ or to be *correct* according to γ. This mimics the situation with the corresponding edits, but note that the set of N-tuples on which a check is "true" is the complement of the corresponding edit.

The following terms from logic will be useful: a *literal* is an atom or a negated ($\neg$) atom; a *clause* is a disjunction ($\vee$) of literals; a clause α *subsumes* a clause β if the literals in α form a subset of the literals in β.

Normal edits are represented by negations of conjunctions ($\wedge$) of clauses, as seen in the Example, part 6. In general, the normal edit $\prod_{j=1}^N A^e_j$ is represented by the check $\neg \bigwedge_{j=1}^N \bigvee_{v \in A^e_j} p^v_j$. These formulae are complex and difficult to reason about, but there are simpler "semantically equivalent" formulae: two formulae φ and ψ are *semantically equivalent* if for all N-tuples R, $f_R(\varphi) = f_R(\psi)$.

Example, part 8. The checks γ_1, γ_2 and γ_3 from Example, part 6, respectively representing the edits e_1, e_2 and e_3, are semantically equivalent to:

$$\begin{array}{lllll} \epsilon_1 = & & p^{\mathrm{N}}_{\mathrm{driver}} & \vee & p^{11}_{\mathrm{grade}} \vee p^{12}_{\mathrm{grade}} \\ \epsilon_2 = p^{10}_{\mathrm{age}} \vee \cdots \vee p^{20}_{\mathrm{age}} & \vee & & & p^{1}_{\mathrm{grade}} \vee \cdots \vee p^{6}_{\mathrm{grade}} \\ \epsilon_3 = p^{10}_{\mathrm{age}} \vee \cdots \vee p^{20}_{\mathrm{age}} & \vee & p^{\mathrm{N}}_{\mathrm{driver}} \,. & & \end{array}$$

In general, the normal edit $e = \prod_{j=1}^{N} A^e_j$ can be represented by the check $\bigvee_{j=1}^{N} \bigvee_{v \in \overline{A^e_j}} p^v_j$, where $\overline{A^e_j}$ is the complement of A^e_j, and so we define a *Fellegi-Holt normal check (FHNC)* ϵ to be a check written in the form $\bigvee_{j=1}^{N} \bigvee_{v \in S^\epsilon_j} p^v_j$, where $S^\epsilon_j \subseteq A_j$. The sets S^ϵ_j may be empty. If they are all empty then the FHNC is the *empty clause*, written $\square$, which is the check according to which no N-tuple is correct, ie every truth function makes $\square$ false.

Given a set Σ of FHNCs, the *FH-deduced check* with generating index i is $\mathrm{FHD}(i, \Sigma)$, an exact mimic of the complement of the $\mathrm{FHG}(i, E)$ in Equation (1):

$$\mathrm{FHD}(i, \Sigma) = \bigvee \left\{ p^v_j \mid j \in \{1, \ldots, N\} \backslash \{i\}, v \in \bigcup_{\epsilon \in \Sigma} S^\epsilon_j \right\} \;\vee\; \bigvee \left\{ p^v_i \mid v \in \bigcap_{\epsilon \in \Sigma} S^\epsilon_i \right\}.$$

Note that $\mathrm{FHD}(i, \Sigma)$ is an FHNC.

Example, part 9. $\mathrm{FHD}(\mathrm{grade}, \{\epsilon_1, \epsilon_2\}) = \epsilon_3$.

As with FH edit generation, any record that satisfies all FHNCs in Σ must also satisfy $\mathrm{FHD}(i, \Sigma)$, so the FH deduction produces acceptable results:

Proposition 2 (Soundness). *Let Σ be a set of FHNCs, i a field, and f a truth function such that for all $\sigma \in \Sigma$, $f(\sigma) = true$. Then $f(\mathrm{FHD}(i, \Sigma)) = true$.*

As with FH edit generation, the FH deduction can be repeatedly applied until no new checks can be generated. We write $\Sigma \vdash_{\mathrm{FH}} \epsilon$ to mean that ϵ is obtained from Σ using one or more steps of FH deduction. As with FH edit generation, checks which represent "dominated" edits can be excluded. This translates to:

1. excluding any subsumed check.
2. excluding any check which is mapped to "true" by every truth function. This will happen if Axiom 2 is part of the check.

We will call the resulting set of FHNCs derived from a starting set Σ of FHNCs the *set of minimal FH-deduced FHNCs*, written $\mathrm{MFH}(\Sigma)$ or just MFH if the context is clear. The field j is an *involved field* of the FHNC $\bigvee_{j=1}^{N} \bigvee_{v \in S^\epsilon_j} p^v_j$ if $S^\epsilon_j \neq \emptyset$. A *covering set* for a record is a set C of fields such that each failed FHNC in the MFH has at least one involved field in C. With these definitions, the error localisation guarantee (Theorem 1) translates to logic as:

Theorem 2 (Translation to logic of Theorem 1, Corollaries 1, 2 of [1], modified in [3]). *Let f be a truth function and let Σ be a set of FHNCs such that $\square \notin \mathrm{MFH}(\Sigma)$. Let $\Sigma' = \{\epsilon \in \mathrm{MFH}(\Sigma) \mid f(\epsilon) = \mathrm{false}\}$ be those FHNCs in $\mathrm{MFH}(\Sigma)$ which f makes false. Then there is a covering set of Σ', and for every covering set C there is a truth function f' such that (1) for all fields j with $j \notin C$ and all $v \in A_j$, $f'(p_j^v) = f(p_j^v)$, and (2) for all $\epsilon \in \Sigma$, $f'(\epsilon) = \mathrm{true}$.*

The proof is a faithful translation to logic of Fellegi and Holt's proof.

5 Using Resolution Deduction instead of FH Deduction

Instead of using Fellegi-Holt deduction, we propose using resolution deduction, explained below. In this section we show that the MFH can be obtained using resolution deduction instead of FH deduction, opening the possibility to use the well-developed methods of automated resolution deduction to obtain the MFH.

Resolution deduction is a well-known method of generating formulae from other formulae [11]. Whereas FH deduction applies to FHNCs, resolution deduction applies to clauses. A single step of resolution takes as input two clauses $q \vee l_1 \vee \cdots \vee l_m$ and $\neg q \vee l'_1 \vee \cdots \vee l'_n$, where q is an atom and $l_1, \ldots, l_m, l'_1, \ldots, l'_n$ are literals, and produces the *resolvent* clause $l_1 \vee \cdots \vee l_m \vee l'_1 \vee \cdots \vee l'_n$, by cancelling out q and $\neg q$. The result of resolving the unit clauses q and $\neg q$ is $\square$.

We define *RA (resolution-axioms)* deduction using resolution and the two Axioms of Section 4. The clause σ is *RA-deducible* from Σ, written $\Sigma \vdash_{\mathrm{RA}} \sigma$, if σ results from $\Sigma \cup \mathrm{Axioms}$ by possibly many resolution steps or $\sigma \in \Sigma \cup \mathrm{Axioms}$.

As with FH-deduction, given a starting set Σ of clauses, we can apply RA-deduction until no new clauses can be generated. The process will eventually terminate because the number of possible clauses is finite. We remove any subsumed clauses or any clause which contains Axiom 2. At the end, we also remove all non-FHNCs. We call the unique resulting set of clauses *the set of minimal RA-deduced FHNCs* and write it as $\mathrm{MRA}(\Sigma)$ or just MRA if the context is clear.

Example, part 10. The FHNC ϵ_3 is RA-deducible from $\{\epsilon_1, \epsilon_2\}$ as follows:

$$\epsilon_1 = p^N_{\mathrm{driver}} \vee p^{11}_{\mathrm{grade}} \vee p^{12}_{\mathrm{grade}} \qquad \epsilon_2 = p^{10}_{\mathrm{age}} \vee \cdots \vee p^{20}_{\mathrm{age}} \vee p^{1}_{\mathrm{grade}} \vee \cdots \vee p^{6}_{\mathrm{grade}}$$

For $v = 1, \ldots, 6$, resolve the axiom $\neg p^{11}_{\mathrm{grade}} \vee \neg p^v_{\mathrm{grade}}$ with ϵ_1 on p^{11}_{grade} to obtain 6 resolvents:

$$p^N_{\mathrm{driver}} \vee p^{12}_{\mathrm{grade}} \vee \neg p^v_{\mathrm{grade}}\,. \qquad (*v)$$

For $v = 1, \ldots, 6$, resolve $(*v)$ with the axiom $\neg p^{12}_{\mathrm{grade}} \vee \neg p^v_{\mathrm{grade}}$ to obtain 6 resolvents:

$$p^N_{\mathrm{driver}} \vee \neg p^v_{\mathrm{grade}}\,. \qquad (+v)$$

Now resolve $(+1)$ with ϵ_2 on p^1_{grade} to obtain:

$$p^{10}_{\mathrm{age}} \vee \cdots \vee p^{20}_{\mathrm{age}} \;\vee\; p^N_{\mathrm{driver}} \;\vee\; p^2_{\mathrm{grade}} \vee \cdots \vee p^6_{\mathrm{grade}}\,. \qquad (\dagger 1)$$

Now resolve (+2) with (†1) on p^2_{grade} to eliminate p^2_{grade} from (†1) to obtain:

$$p^{10}_{\text{age}} \vee \cdots \vee p^{20}_{\text{age}} \ \vee \ p^{N}_{\text{driver}} \ \vee \ p^{3}_{\text{grade}} \vee \cdots \vee p^{6}_{\text{grade}} \,. \qquad (\dagger 2)$$

Continue successively resolving on $p^3_{\text{grade}}, \ldots, p^6_{\text{grade}}$ to eventually obtain

$$\epsilon_3 = p^{10}_{\text{age}} \vee \cdots \vee p^{20}_{\text{age}} \ \vee \ p^{N}_{\text{driver}} \,. \qquad (\dagger 6)$$

Example part 10 can be generalised on the number and sizes of fields. Some induction steps yield $\text{MFH}(\Sigma) \subseteq \text{MRA}(\Sigma)$. The reverse direction, $\text{MRA}(\Sigma) \subseteq \text{MFH}(\Sigma)$, requires induction on the number of resolution steps, yielding:

Theorem 3. *If Σ is a set of FHNCs then MFH(Σ) = MRA(Σ).*

Theorem 3 has two consequences. Firstly, automated resolution deduction will find the MFH. Secondly, the set of MRAs guarantees error localisation since Theorem 2 applies, and as with FH-deduction, a smallest covering set is guaranteed to be a solution to the error localisation problem.

The Fellegi-Holt theorems in logical perspective. In effect Fellegi and Holt proved some well-known and important results about resolution [11]. Their Lemma 1 is equivalent to the standard logic result known as soundness, and their Theorem 1 has as a simple consequence the standard logic result known as refutational completeness. In fact, their Theorem 1 (part of our Theorem 2) is a generalisation of a standard logic lemma [11, Lemma 8.14, page 55]. Although we have omitted the details, our logical equivalents of the Fellegi-Holt theorems show that, contrary to the claim of Fellegi and Holt, their Theorem 1 can be proved without using their Theorem 2. These observations support our claim that a recasting of the Fellegi-Holt method into formal logic helps to reason about the method itself.

6 Experiments

We conducted experiments to prove our claim that FH-generation can be done using resolution. We chose OTTER [12], a resolution-based theorem prover, for its reliability and easy availability. Since we were only testing our claim, it did not matter that OTTER is not tuned for propositional deduction. We used OTTER in its default configuration, on a Sun Ultra Sparc 250, and used hyper-resolution (a special type of resolution) as the deduction method, and the "propositional" setting. The clause lists were as follows: "usable" list - the given checks; "set of support" list - Axiom 1; "passive" list - Axiom 2. We used OTTER to generate the MRA for various initial sets of checks.

Our initial tests verified that OTTER does in fact find all the required deduced checks. In particular we used the example analysed in [5] and [6], and obtained the same deduced checks. The CPU time for this small example was 0.7 seconds.

It is hard to compare experimental results since any other implemetations are likely to be out of date, because the FH method has not been used in statistical agencies recently. The only published results that we know of are in [5].

We conducted tests to assess the effect on CPU time of increasing the initial number of checks. We used various combinations of three parameters: number of checks, number of fields and number of values per field. For each such combination we produced 1000–3000 randomly generated sets of edits, in order to observe the variability of the results. Our sets of checks are arranged in two groups, Group A and Group B, whose parameters are listed in Table 1. The sets of checks in Group A contained checks with 13 fields and up to 8 values per field. In Group B, the sets of checks contained checks with 10 fields and up to 10 values per field. Each Group contained sets of checks with 14, 19, 28 and 35 checks. Table 2 gives the CPU times of our results, listing the median times, and the upper 75% and 99% percentiles.

Table 1. Experiment parameters

	Group A	Group B
No. of fields	13	10
No. of values per field	Fields 1 & 2: 2 Fields 3 & 4: 3 Fields 5 & 6: 4 Fields 7 & 8: 5 Fields 9 & 10: 6 Fields 11 & 12: 7 Field 13: 8	Fields 1 & 2: 2 Field 3: 3 Field 4: 4 Field 5: 5 Field 6: 6 Field 7: 7 Field 8: 8 Field 9: 9 Field 10: 10

Table 2. CPU time (secs) to generate MRAs

	Group A			Group B		
Initial no. of FHNCs	Median	75% percentile	99% percentile	Median	75% percentile	99% percentile
14	1	1	39	2	5	147
19	2	6	280	3	14	898
28	2	12	2297	1	5	904
35	1	4	1049	1	2	165

As the initial number of checks increases, the computational time first increases and then decreases. The decrease is caused by the interconnectedness of the checks, that is, the checks are more likely to have fields in common which

are then eliminated in the deduction process. For example, as the number of randomly generated checks increases, an inconsistency becomes more likely, reducing the MRA to the single clause $\square$. In addition, clauses restricted to one field are also more likely, so that newly generated clauses are more likely to be subsumed. Such situations are unrealistic: contradictory checks are almost always unacceptible as are checks which unduly restrict one field. Thus these experiments demonstrate some characteristics of randomly generated problems as opposed to real problems. The computational times decline because of the increasing proportion of unrealistic sets of checks.

Efficient methods for performing deduction have been studied extensively in the field of automated deduction, and OTTER is by no means state-of-the-art. Systems like Chaff [13] can now perform automated deduction in propositional logic orders of magnitude faster than can OTTER. Simon and del Val [14] state that modern consequence finders can generate and represent huge numbers (10^{70}) of deduced clauses efficiently. With such systems, one would expect that the CPU times will be much lower.

7 Conclusions and Future Work

We have shown the equivalence between Fellegi-Holt deduction and resolution deduction. Fellegi and Holt's work is truly impressive: in apparent isolation from logicians, they reinvented automated propositional deduction and proved the underlying theorems.

The difficulty with the Fellegi-Holt method is the slowness of generating a suitable set of edits for error localisation. As noted by Bruni and Sassano [9], methods like Fellegi-Holt, which generate all implied edits, are not applicable to many problems because the number of implied edits increases exponentially with the number of original edits. The translation to propositional logic cannot avoid this problem, of course, since it is inherent in any NP-complete problem. But modern consequence finders [14] and propositional satisfiability solvers [13] can efficiently handle huge problems which also have the same exponential behaviour. So they are worth investigating for our fomalisation of the Fellegi-Holt method.

We have shown that logic has potential for improving the method by giving a rigorous framework for reasoning about the method, and by demonstrating that the core of the Fellegi-Holt method can be implemented in automated deduction tools based on logic. Our next step will be to use some well-tuned deduction system, or consequence finder, to generate checks efficiently.

Acknowledgements. Thank you to the three anonymous referees for their helpful suggestions, to Matt Gray for his efficient programming assistance, and to Andrew Slater for informative discussions about propositional satisfiability solvers.

References

1. Fellegi, I.P., Holt, D.: A systematic approach to automatic edit and imputation. Journal of the American Statistical Association **71** (1976) 17–35
2. U.N. Statistical Commission, Economic Commission for Europe and Conference of European Statisticians: Work Session on Statistical Data Editing. Helsinki (2002)
3. Liepins, G.E.: A rigorous, systematic approach to automatic editing and its statistical basis. Report ORNL/TM-7126, Oak Ridge National Lab., Tennessee (1980)
4. Liepins, G.E.: Refinements to the Boolean approach to automatic data editing. Technical Report ORNL/TM-7156, Oak Ridge National Lab., Tennessee (1980)
5. Garfinkel, R.S., Kunnathur, A.S., Liepins, G.E.: Optimal imputation of erroneous data: Categorical data, general edits. Operations Research **34** (1986) 744–751
6. Winkler, W.E.: Editing discrete data. Statistical Research Report Series, RR97/04, U.S. Bureau of the Census (1997)
7. Winkler, W.E., Chen, B.C.: Extending the Fellegi-Holt model of statistical data editing. Research Report Series, Statistics #2002-02, U.S. Bureau of the Census (2002)
8. Barcaroli, G.: Un approccio logico formale al probleme del controllo e della correzione dei dati statistici. Quaderni di Ricerca 9/1993, Istituto Nazionale di Statistica , Italia (1993)
9. Bruni, R., Sassano, A.: Errors detection and correction in large scale data collecting. In: Advances in Intelligent Data Analysis, Volume 2189 of Lecture Notes in Computer Science, Springer-Verlag (2001) 84–94
10. Franconi, E., Palma, A.L., Leone, N., Perri, S., Scarcello, F.: Census data repair: A challenging application of disjunctive logic programming. In: 8th International Conference on Logic for Programming, Artificial Intelligence and Reasoning (2001)
11. Nerode, A., Shore, R.A.: Logic for Applications. Springer-Verlag, New York (1997)
12. Argonne National Laboratory, Math. & Comp. Sci. Div.: Otter: An automated deduction system. (Web page) http://www-unix.mcs.anl.gov/AR/otter/
13. SAT Research Group, Electrical Engineering Department, Princeton University: zChaff. (Web page) http://ee.princeton.edu/~chaff/zchaff.php
14. Simon, L., del Val, A.: Efficient consequence finding. In: IJCAI '01: 17th International Joint Conference on Artificial Intelligence, Morgan Kaufmann (2001)