Chapter 2

The covering set method

2.1 Introduction

In Chapter 1 we observed the idea of using "covering sets" to solve the error localisation problem. In this chapter we formally define the technique of using covering sets, called the "covering set method" (Definition 2.3.12), supported by the underpinning definitions and some basic properties.

Since the covering set method is used to solve the error localisation problem, we first define the error localisation problem, as well as the broader error correction problem. Both the error correction problem and the error localisation problem are defined in the next section (in Definitions 2.2.6 and 2.2.8 respectively). Having defined the error localisation problem, we will be ready to formally define the covering set method, which we do in the subsequent section, Section 2.3.

As seen in Chapter 1, the method of using covering sets does not always work in the sense that it does not always succeed in solving the error localisation problem. But it does work when certain "edit generation" functions (Definition 2.3.13) are used within the method. The method works when the relevant edit generation function has a property called "covering set correctibility" (Definition 2.3.14).

I shall present several functions that are covering set correctible, in Sections 2.4 to 2.6. I start, in Section 2.4, with the function FH, originally defined by Fellegi and Holt (1973, 1976). The functions presented in the subsequent two sections (Sections 2.5 and 2.6) are all subfunctions of FH. They are called ENFH, MFH, MENFH and FCF_ω. I also present the proofs of the covering set correctibility of the various functions, although the proof for FCF_ω is complicated and deferred until Chapter 6.

There are also some techniques to speed up the covering set method and to reduce its memory usage. Some of the techniques tinker with the edit generation functions, while others are compromises to the method. One technique that has not been much explored is the use of logic, the topic of this thesis. In Section 2.7 I discuss some of the practicalities of using the covering set method.

This chapter is mainly a summary of material previously published by other authors. However there are some aspects that are new, or published in some of my previous papers (Boskovitz et al. 2005; Boskovitz and Goré 2005).

The first new aspect is the property of covering set correctibility. Other authors

have implicitly used a component of covering set correctibility, which I call the error correction guarantee (Definition 2.3.16). Garfinkel et al. (1986b) have used a property called sufficiency, which is similar to the error correction guarantee. But I am not aware of any authors using the other component of covering set correctibility, which I call error correction totality (Definition 2.3.16).

The second new aspect concerns the proofs of covering set correctibility for the various functions. The proofs of one component, the error correction totality, are new, although Fellegi and Holt (1973, 1976) did prove the main result needed to prove error correction totality. As to the other component, the error correction guarantee, the situation is more complicated. For the functions other than FCF_ω, the proofs have been previously published in full or in essence. However some of the results are special cases of more general results about superior functions and subfunctions, and the proofs presented here are in those terms.

This chapter focuses on categorical edits, rather than arithmetic edits. The corresponding definitions and properties for arithmetic edits are presented in Chapter 7.

We start with the background needed to define the covering set method, namely the definition of the error localisation problem.

2.2 Error correction and error localisation

In this section, we define the problem solved by the covering set method. We also define the various components of the problem. The covering set method solves the error localisation problem, to be defined in Definition 2.2.8, which is part of a technique to solve the error correction problem, to be defined in Definition 2.2.6. Hence we define the error correction problem and its components.

We assume that we are dealing with data arranged in records, which are N-tuples, where N is fixed for all records under consideration.

Definition 2.2.1. *A **record** $\boldsymbol{v}$ is an N-tuple $(v_1, \ldots, v_N)$ where N is a positive integer; $j = 1, \ldots, N$ are called **fields**; and $v_j \in A_j$ where each A_j is a set containing at least two elements, called the jth **field domain**. We assume that N is the same for all records. The set of all possible records is the **data domain** $D = A_1 \times \cdots \times A_N$ (also called the **domain**).*

Example 2.2.2. *A data table about 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. The data table includes three fields, with field domains listed below:*

A_1 (relevant ages), which we relabel $A_{\mathsf{age}} = \{5, 6, \ldots, 20\}$;
A_2 (whether someone has a driver's licence), which we relabel $A_{\mathsf{driver}} = \{\mathrm{yes}, \mathrm{no}\}$;
A_3 (school grade levels), which we relabel $A_{\mathsf{grade}} = \{1, 2, \ldots, 12\}$.

The data domain D is $A_{\mathsf{age}} \times A_{\mathsf{driver}} \times A_{\mathsf{grade}}$.
The record in this example is $\boldsymbol{vv} = (6, \mathrm{yes}, 8)$.

Edits (defined precisely below) are used to specify potential errors in each record. Edits apply to one record at a time. We define categorical edits below, while we leave the definition of arithmetic edits to Chapter 7. Each categorical edit specifies a rejection region, that is, a set of unacceptable records. More formally we have:

Definition 2.2.3. *A* ***categorical edit*** *is a subset of* $D = A_1 \times \cdots \times A_N$.

A record $\boldsymbol{v}$ *is* ***incorrect*** *with respect to the categorical edit* e *if* $\boldsymbol{v} \in e$*. We can also say that* $\boldsymbol{v}$ ***fails*** e*. If* $\boldsymbol{v} \notin e$*, we say that* $\boldsymbol{v}$ *is* ***correct*** *with respect to* e*, or* $\boldsymbol{v}$ ***satisfies*** e*. The record* $\boldsymbol{v}$ *fails a set* E *of edits if* $\boldsymbol{v}$ *fails some edit in* E*. The record* $\boldsymbol{v}$ *satisfies a set* E *of edits if* $\boldsymbol{v}$ *satisfies all edits in* E*.*

Note: The set D is an edit, although it is failed by every record.

Example 2.2.4. *Suppose that the following requirements apply to the data table of Example 2.2.2:*

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 edits e_1 *and* e_2 *corresponding to the above two requirements are the subsets of* D *that fail each respective requirement:*

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

We will write $EE = \{e_1, e_2\}$*.*
Consider the record $\boldsymbol{vv} = (6, \text{yes}, 8)$*. Since* $\boldsymbol{vv} \in e_1$ *and* $\boldsymbol{vv} \in e_2$ *, we say that* $\boldsymbol{vv}$ *fails both edits.*

We will use the term "dominate" to describe the strict superset relationship between edits:

Definition 2.2.5. *(Kunnathur 1982) Edit* e ***dominates*** *edit* e' *if* $e \supset e'$*.*

Having defined edits and records, we can now formally define the data correction problem:

Definition 2.2.6. *Given a record* $\boldsymbol{v}$ *and a set* E *of edits, the* ***error correction problem*** *is the problem of choosing a suitable record* $\boldsymbol{w}$ *that satisfies each edit in* E*. We say that* ***the record*** $\boldsymbol{w}$ ***corrects*** $\boldsymbol{v}$ ***with respect to*** E*.*

Example 2.2.7. *As in Examples 2.2.2 and 2.2.4, suppose that*

$$\begin{array}{rcccccccl} D & = & A_{\mathsf{age}} & \times & A_{\mathsf{driver}} & \times & A_{\mathsf{grade}}\,, & & \\ EE & = & \{e_1, e_2\}, \textit{ where} & & & & & & \\ e_1 & = & A_{\mathsf{age}} & \times & \{\text{yes}\} & \times & \{1,2,\ldots,10\}, & & \\ e_2 & = & \{5,6,\ldots,9\} & \times & A_{\mathsf{driver}} & \times & \{7,8,\ldots,12\}, & & \textit{and} \\ \boldsymbol{vv} & = & (6, \text{yes}, 8), \textit{ which fails both edits } e_1 \textit{ and } e_2\,. & & & & & & \end{array}$$

Since the record $\boldsymbol{ww} = (6, \text{no}, 1)$ *satisfies both edits, we say that* $\boldsymbol{ww}$ *corrects the record* $\boldsymbol{vv}$ *with respect to the edit set* EE.

How to choose a "suitable" record $\boldsymbol{w}$? This thesis deals with one way: use the covering set method to decide which fields should be changed (error localisation), and then impute on those fields. Hence we give a formal definition of the error localisation problem:

Definition 2.2.8. *Let* $\boldsymbol{v} = (v_1, \ldots, v_N)$ *be a record and* E *be a set of edits. The* ***error localisation problem*** *is the problem of deciding which sets of fields can be changed to correct* $\boldsymbol{v}$ *with respect to* E. *That is, it is the following problem: given a set* C *of fields and a record* $\boldsymbol{v}$, *decide whether there is a record* $\boldsymbol{w} = (w_1, \ldots, w_N)$ *such that*

a. $C \supseteq \{j \mid w_j \neq v_j\}$, *and*

b. $\boldsymbol{w}$ *satisfies all edits in* E.

If there exists a $\boldsymbol{w}$ *such that a and b are satisfied, then we say that* ***the set*** C ***yields a correction*** $\boldsymbol{w}$ ***to*** $\boldsymbol{v}$ ***with respect to*** E *and we shall also say that that* C ***is a solution to the error localisation problem for*** $\boldsymbol{v}$ ***and*** E.

The set of solutions to the error localisation problem for the record $\boldsymbol{v}$ *and the edit set* E *will be written* $\mathcal{EL}(E, \boldsymbol{v})$. *This means that* C *yields a correction to* $\boldsymbol{v}$ *with respect to* E *if and only if* $C \in \mathcal{EL}(E, \boldsymbol{v})$.

Note 1: $\boldsymbol{v}$ satisfies E if and only if $\emptyset \in \mathcal{EL}(E, \boldsymbol{v})$.
Note 2: If $D \in E$ then $\mathcal{EL}(E, \boldsymbol{v}) = \emptyset$.

Example 2.2.9. *In Example 2.2.7, the correct record* $\boldsymbol{ww}$ *differs from the incorrect record* $\boldsymbol{vv}$ *on the fields* driver *and* grade. *We say that the field set* {driver, grade} *yields the correction* $\boldsymbol{ww}$, *among others, to* $\boldsymbol{vv}$ *with respect to* EE. *It is impossible to correct* $\boldsymbol{vv}$ *by changing just one field, but any pair of fields is an error localisation solution for* $\boldsymbol{vv}$ *with respect to* EE, *as is the set of all three fields. Thus*

$$\mathcal{EL}(EE, \boldsymbol{vv}) = \big\{\{\mathsf{age}, \mathsf{driver}\}, \{\mathsf{age}, \mathsf{grade}\}, \{\mathsf{driver}, \mathsf{grade}\}, \{\mathsf{age}, \mathsf{driver}, \mathsf{grade}\}\big\}.$$

In practice we want to solve a tighter problem than the above error localisation problem. We are only interested in the "best" field sets C which we choose according to some criterion. A suitable criterion is to seek a smallest set of weighted fields that can be changed to correct $\boldsymbol{v}$, as used by Garfinkel et al. (1986b). We call this problem the "smallest weighted error localisation" problem and define it as follows.

Definition 2.2.10. *The* ***smallest weighted error localisation problem*** *is the following problem:*

Given an edit set E, *a record* $\boldsymbol{v} = (v_1, \ldots, v_N)$, *and a vector* $\boldsymbol{b} = (b_1, \ldots, b_N)$ *of* N *positive numbers (weights), decide whether the field set* C *minimises* $\sum_{i \in C} b_i$ *such that the set* C *yields a correction to* $\boldsymbol{v}$ *with respect to* E.

The set of solutions to the smallest weighted error localisation problem for record $\boldsymbol{v}$ *with respect to edit set* E *and weights vector* $\boldsymbol{b}$ *will be written* $\mathcal{SEL}(E, \boldsymbol{v}, \boldsymbol{b})$.

Note 1: Each solution to the smallest weighted error localisation problem is a weighted example of a "cardinality-oriented repair" defined by Bertossi et al. (2003, Definition 2(c)).

Note 2: Another criterion for choosing the "best" field set C is to choose a set C that yields a correction and that is minimal under set inclusion. Bertossi et al. (2003, Definition 2(b)) give the name "subset-oriented repair" to any correction obtained by such a criterion.

Example 2.2.11. *As in the previous examples, suppose that*

$$\begin{array}{rcl} D & = & A_{\mathsf{age}} \times A_{\mathsf{driver}} \times A_{\mathsf{grade}}\,, \\ EE & = & \{e_1, e_2\}, \text{ where} \\ e_1 & = & A_{\mathsf{age}} \times \{\text{yes}\} \times \{1, 2, \ldots, 10\}, \\ e_2 & = & \{5, 6, \ldots, 9\} \times A_{\mathsf{driver}} \times \{7, 8, \ldots, 12\}, \quad \text{and} \\ \boldsymbol{vv} & = & (6, \text{yes}, 8), \text{ which fails both edits } e_1 \text{ and } e_2\,. \end{array}$$

As seen in Example 2.2.9, the record $\boldsymbol{vv}$ *can be corrected by suitable changes to any pair of fields, but cannot be corrected by changing only one field. If each field is weighted equally, that is* $\boldsymbol{b} = (1, \ldots, 1)$*, then* $\mathcal{SEL}\big(EE, \boldsymbol{vv}, (1, \ldots, 1)\big)$ *is the set of all pairs of fields:*

$$\mathcal{SEL}\big(EE, \boldsymbol{vv}, (1, \ldots, 1)\big) = \big\{\{\mathsf{age}, \mathsf{driver}\}, \{\mathsf{age}, \mathsf{grade}\}, \{\mathsf{driver}, \mathsf{grade}\}\big\}.$$

It turns out that the error localisation problem is more easily addressed if all the edits are of normal form, defined next.

Definition 2.2.12. *A non-empty categorical edit* e *is of* ***normal form*** *if* $e = A_1^e \times \cdots \times A_N^e$ *where for each* $j = 1, \ldots, N$*, the set* $A_j^e \subseteq A_j$ *.*

The ***normal form of the empty edit*** *is* $\emptyset \times \cdots \times \emptyset$*, with* $\emptyset$ *repeated* N *times. The other representations of the empty edit as products are not considered to be of normal form.*

Instead of the term "an edit of normal form" we also use the term ***normal edit****.*

If $X \subseteq D$ *then we write* $\mathcal{N}(X)$ *to mean the set of normal edits that are subsets of* X*. The set* $\mathcal{N}(X)$ *is a subset of the power set* $\mathcal{P}(X)$*.*

Notation. *For a normal edit* e*, we will use the notation of Definition 2.2.12. That is* $e = A_1^e \times \cdots \times A_N^e$ *where for each* $j = 1, \ldots, N$*, the set* $A_j^e \subseteq A_j$ *.*

Note: Later on, we will see that the most useful elements of $\mathcal{N}(X)$ are the maximal normal edits contained in X.

Example 2.2.13. *The edits* e_1 *and* e_2 *are presented in normal form in Example 2.2.11. An example of a non-normal edit is* $\big\{(6, \text{yes}, 8), (7, \text{no}, 12)\big\}$*, which cannot be written as a product of subsets of the fields.*

In general we will assume that all categorical edits are in normal form, because the next proposition tells us that each categorical edit can be broken down as a union of a set of normal edits.

Proposition 2.2.14. *If E is a set of edits (some of which might not be normal) then there exists a set E' of normal edits such that, for each record $\boldsymbol{v}$, the record $\boldsymbol{v}$ fails E if and only if $\boldsymbol{v}$ fails E'.*

Proof. At worst $E' = \big\{\{a_1\} \times \cdots \times \{a_N\} \mid (a_1, \ldots, a_N)$ is an element of some edit in $E\big\}$. ⊣

We will occasionally use the following property of normal edits.

Lemma 2.2.15. *If e_1 and e_2 are normal edits with $e_1 \subseteq e_2$, then for each field j, we have that $A_j^{e_1} \subseteq A_j^{e_2}$.*

Proof. If $e_1 \neq \emptyset$, then the result follows from a property of sets. If $e_1 = \emptyset$, then the result follows from the definition of the normal form for the empty edit as the product of N empty sets. ⊣

2.3 Solving the error localisation problem using covering sets

This section gives a formal definition of covering sets (Definition 2.3.3) and of the covering set method (Definition 2.3.12). The covering set method depends on the definition of edit generation functions (Definition 2.3.13) which can have various properties: covering set correctibility (Definition 2.3.14), error correction guarantee (Definition 2.3.16), error correction totality (Definition 2.3.16), and smallest weighted covering set correctibility (Definition 2.3.19).

The covering set method depends on the following observations.

Observation 1. Suppose that the record $\boldsymbol{v}$ fails the normal edit e. Then any correction to $\boldsymbol{v}$ with respect to $\{e\}$ must change at least one field j for which A_j^e is a strict subset of A_j, because otherwise the change in field j cannot change $\boldsymbol{v}$'s correctness with respect to e.

Observation 2. Hence if $\boldsymbol{v}$ fails a set E of normal edits, then any correction to $\boldsymbol{v}$ changes a set C of fields where to each failed edit e in E there is a field j in C such that A_j^e is a strict subset of A_j. Such a set C is a solution to the error localisation problem for $\boldsymbol{v}$ and E.

Example 2.3.1. *As in the previous examples, suppose that*

$$\begin{array}{rcl}
D & = & A_{\mathsf{age}} \times A_{\mathsf{driver}} \times A_{\mathsf{grade}}, \\
EE & = & \{e_1, e_2\}, \textit{ where} \\
e_1 & = & A_{\mathsf{age}} \times \{\text{yes}\} \times \{1, 2, \ldots, 10\}, \\
e_2 & = & \{5, 6, \ldots, 9\} \times A_{\mathsf{driver}} \times \{7, 8, \ldots, 12\}, \\
\boldsymbol{vv} & = & (6, \text{yes}, 8), \textit{ which fails both edits } e_1 \textit{ and } e_2, \textit{ and} \\
\boldsymbol{ww} & = & (6, \text{no}, 1), \textit{ which corrects } \boldsymbol{vv} \textit{ with respect to the edit set } EE.
\end{array}$$

The correction $\boldsymbol{ww}$ *changes the values of the fields of* $\boldsymbol{vv}$ *in the set* $\{\mathsf{driver}, \mathsf{grade}\}$. *Thus Observation 2 applies to* EE, $\boldsymbol{vv}$ *and* $\boldsymbol{ww}$ *because:*

for edit e_1 *of* EE, *the sets* $A^{e_1}_{\mathsf{driver}} = \{\text{yes}\}$ *and* $A^{e_1}_{\mathsf{grade}} = \{1, 2, \ldots, 10\}$ *are strict subsets of* A_{driver} *and* A_{grade} *respectively; and*

for edit e_2 *of* EE, *the set* $A^{e_2}_{\mathsf{grade}} = \{7, 8, \ldots, 12\}$ *is a strict subset of* A_{grade}.

We call the set C of Observation 2 a "covering set of the edits of E failed by $\boldsymbol{v}$" and we say that a field j with the property that A^e_j is a strict subset of A_j is "involved in e". More formally:

Definition 2.3.2. *The field* j *is* ***involved*** *or* ***entering*** *in edit* $e = A^e_1 \times \cdots \times A^e_N$ *if* $A^e_j \neq A_j$. *An* ***uninvolved*** *or* ***non-entering*** *field* j *has* $A^e_j = A_j$ *and is also referred to as an* ***essential*** *field.*

Note: Every field is involved in the empty edit.

Definition 2.3.3. *A* ***covering set*** C *of the set* E *of edits is a set of fields such that, for each edit* e *in* E, *there exists a field in* C *that is involved in* e. *The set of all covering sets of* E *is written* $\mathcal{C}(E)$:

$$\mathcal{C}(E) = \big\{C \subseteq \{1, \ldots, N\} \mid C \text{ is a covering set of } E\big\}.$$

That is, a covering set of the edit set E is a set of fields that includes an involved field of each edit in E.

Note 1: If $C \in \mathcal{C}(E)$ then every superset of C is also in $\mathcal{C}(E)$.

Note 2: $E = \emptyset$ if and only if $\emptyset \in \mathcal{C}(E)$. Hence, by Note 1, any set of fields covers the edit set $\emptyset$.

Note 3: $D \in E$ if and only if $\mathcal{C}(E) = \emptyset$.

Example 2.3.4. *In the previous examples,*

the edit e_1 *involves the fields* driver *and* grade*;*

the edit e_2 *involves the fields* age *and* grade.

The two edits are covered by the singleton field set $\{\mathsf{grade}\}$ *and also by any field set that includes* grade*; they are also covered by the doubleton field set* $\{\mathsf{age}, \mathsf{driver}\}$.

We are especially interested in finding a covering set of the edits of E failed by $\boldsymbol{v}$, and so the following definition will be useful.

Definition 2.3.5. *Define the set* $\mathcal{X}(E, \boldsymbol{v})$ *to be the set of edits in the set* E *that are failed by* $\boldsymbol{v}$. *That is,* $\mathcal{X}(E, \boldsymbol{v}) = \{e \in E \mid \boldsymbol{v} \in e\}$.

By Definitions 2.3.3 and 2.3.5, the set of covering sets of the edits of E failed by $\boldsymbol{v}$ is $\mathcal{C}\big(\mathcal{X}(E, \boldsymbol{v})\big)$. Since we will be using the combination of $\mathcal{C}$ and $\mathcal{X}$ often, we will simplify notation a little, as follows.

Definition 2.3.6. *Define* $\mathcal{CX}(E, \boldsymbol{v})$ *to be the set of covering sets of the edits in the set* E *that are failed by the record* $\boldsymbol{v}$. *That is* $\mathcal{CX}(E, \boldsymbol{v}) = \mathcal{C}\big(\mathcal{X}(E, \boldsymbol{v})\big)$.

Example 2.3.7. *In the previous examples, we found that* $\boldsymbol{vv}$ *fails both edits of* EE*. Hence* $\mathcal{X}(EE, \boldsymbol{vv}) = EE$*. From Example 2.3.4, we have that*

$$\mathcal{CX}(EE, \boldsymbol{vv}) = \big\{\{\mathsf{grade}\}, \{\mathsf{age}, \mathsf{driver}\}, \{\mathsf{age}, \mathsf{grade}\}, \{\mathsf{driver}, \mathsf{grade}\}, \{\mathsf{age}, \mathsf{driver}, \mathsf{grade}\}\big\}.$$

Although every error localisation solution is a covering set of the failed edits, we saw in Chapter 1 that the converse need not apply. That is, there is no guarantee that every covering set of the failed edits is an error localisation solution, and so the covering sets do not solve the error localisation problem. In terms of notation:

$$\text{although } \mathcal{EL}(E, \boldsymbol{v}) \subseteq \mathcal{CX}(E, \boldsymbol{v}), \tag{2.3.1}$$

$$\text{there is no guarantee that } \mathcal{EL}(E, \boldsymbol{v}) = \mathcal{CX}(E, \boldsymbol{v}). \tag{2.3.2}$$

Example 2.3.8. *Defining the edit set* EE *and the record* $\boldsymbol{vv}$ *as in the previous examples, we see from Examples 2.2.9 and 2.3.7 that* $\mathcal{CX}(EE, \boldsymbol{vv}) \neq \mathcal{EL}(EE, \boldsymbol{vv})$*. In particular,* $\{\mathsf{grade}\} \in \mathcal{CX}(EE, \boldsymbol{vv})$*, but* $\{\mathsf{grade}\} \notin \mathcal{EL}(EE, \boldsymbol{vv})$*.*

As observed in Chapter 1, the reason that $\mathcal{EL}(E, \boldsymbol{v})$ need not equal $\mathcal{CX}(E, \boldsymbol{v})$ is that there can be additional edits implied by E which are not covered by every element of $\mathcal{CX}(E, \boldsymbol{v})$ but which are failed by $\boldsymbol{v}$.

Example 2.3.9. *As seen in Example 2.2.4, the edits*

$$\begin{array}{lclclclcl} e_1 & = & A_{\mathsf{age}} & \times & \{\text{yes}\} & \times & \{1, 2, \ldots, 10\} & \textit{and} \\ e_2 & = & \{5, 6, \ldots, 9\} & \times & A_{\mathsf{driver}} & \times & \{7, 8, \ldots, 12\} & \end{array}$$

correspond to the requirements

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

which imply another requirement:

Requirement 3: *A person with a driver's licence must be at least 10 years old. (Of course, in reality, something much stronger than this holds, but this requirement is a logical implication of the first two requirements.)*

Requirement 3 corresponds to the edit

$$e_3 = \{5, 6, \ldots, 9\} \times \{\text{yes}\} \times A_{\mathsf{grade}}\,.$$

Although the field set $\{\mathsf{grade}\}$ *covers the edit set* $\{e_1, e_2\}$*, it does not cover the implied edit set* $\{e_1, e_2, e_3\}$*.*

Although Statement 2.3.2 tells us that in general $\mathcal{EL}(E, \boldsymbol{v}) \neq \mathcal{CX}(E, \boldsymbol{v})$, Fellegi and Holt (1973, 1976) used the idea of implied edits to find a similar, but correct,

relationship. They constructed, for each edit set E, an edit set $\mathrm{FH}(E)$ such that the covering sets of the failed edits of the new set $\mathrm{FH}(E)$ are exactly the error localisation solutions. In other words, the incorrect equation (in Statement 2.3.2) can be modified to give the correct equation

$$\mathcal{EL}(E, \boldsymbol{v}) = \mathcal{CX}\big(\mathrm{FH}(E), \boldsymbol{v}\big).$$

Fellegi and Holt's construction for $\mathrm{FH}(E)$ will be given in the next section, but the following example gives $\mathrm{FH}(EE)$ for the edit set EE of the examples.

Example 2.3.10. *Suppose that the edit set $EE = \{e_1, e_2\}$ is as defined in the previous examples. Then $\mathrm{FH}(EE) = \{e_1, e_2, \ldots, e_8\}$, where e_1, e_2 and e_3 are as given in Example 2.3.9, and*

$$\begin{array}{rcccccc}
e_4 & = & A_{\mathsf{age}} & \times & \{\text{yes}\} & \times & \{7,8,9,10\}, \\
e_5 & = & \{5,6,\ldots,9\} & \times & A_{\mathsf{driver}} & \times & \{7,8,9,10\}, \\
e_6 & = & \{5,6,\ldots,9\} & \times & \{\text{yes}\} & \times & \{7,8,9,10\}, \\
e_7 & = & \{5,6,\ldots,9\} & \times & \{\text{yes}\} & \times & \{1,2,\ldots,10\}, \\
e_8 & = & \{5,6,\ldots,9\} & \times & \{\text{yes}\} & \times & \{7,8,\ldots,12\}.
\end{array}$$

Then $\mathcal{CX}\big(\mathrm{FH}(EE), \boldsymbol{vv}\big) = \mathcal{EL}(EE, \boldsymbol{vv})$
$= \big\{\{\mathsf{age}, \mathsf{driver}\}, \{\mathsf{age}, \mathsf{grade}\}, \{\mathsf{driver}, \mathsf{grade}\}, \{\mathsf{age}, \mathsf{driver}, \mathsf{grade}\}\big\}$.

In the above example, we started with the edits e_1 and e_2, from which we derived the remaining edits $e_3, \ldots, e_8$. We will call the starting set of edits "explicit edits":

Definition 2.3.11. *The starting set of edits, obtained from domain knowledge, is called the set of* ***explicit edits****, to distinguish them from implied edits.*

We will use the term "covering set method" to refer to the technique of using covering sets for error localisation, and define it precisely below.

Definition 2.3.12. *The* ***covering set method*** *is a method of trying to solve the error localisation problem. Given a normal edit set E, the method consists of the following steps:*

1. *Calculate a normal edit set $G(E)$ derived from the set E, where G is some predefined function.*

2. *For any record $\boldsymbol{v}$ that fails E, calculate the field set $\mathcal{CX}\big(G(E), \boldsymbol{v}\big)$ of the covering sets of the edits of $G(E)$ failed by $\boldsymbol{v}$.*

Note that the covering set method does not necessarily work, that is, the calculated field set $\mathcal{CX}\big(G(E), \boldsymbol{v}\big)$ need not be equal to the set of error localisation solutions. For example if $G(E) = E$ then, as seen in Example 2.3.8, the covering set method need not work. On the other hand, if $G(E) = \mathrm{FH}(E)$ then we will see in the next section that the covering set method does work, and that the set $\mathcal{CX}\big(\mathrm{FH}(E), \boldsymbol{v}\big)$ is indeed the set of error localisation solutions.

The covering set method is often referred to as the Fellegi-Holt method, although sometimes the Fellegi-Holt method refers explicitly to using the covering set method with the particular function FH. Occasionally the term "Fellegi-Holt method" is also used for some or all of the criteria specified by Fellegi and Holt (1973, 1976), some of which have been alluded to previously:

1. The (weighted) minimum number of fields should be changed (where the word "weighted" was added by Garfinkel (1979) and Liepins (1980b)) - see also page 6 of Chapter 1.

2. Imputation rules should derive automatically from the edit rules.

3. Imputation should try to maintain the marginal and preferably the joint frequency distributions of variables - see also page 6 of Chapter 1.

The terms "Fellegi-Holt paradigm" and "Fellegi-Holt principle" are also used for one or more of the above criteria.

The function FH, and the function G used in Definition 2.3.12, are described as "normal edit generation functions", defined next.

Definition 2.3.13. *An **edit generation function** is a function from edit sets to edit sets, that is, a function G where*

$$G : \mathcal{P}\big(\mathcal{P}(D)\big) \longrightarrow \mathcal{P}\big(\mathcal{P}(D)\big).$$

*A **normal edit generation function** is a function from normal edit sets to normal edit sets, that is, a function G where*

$$G : \mathcal{P}\big(\mathcal{N}(D)\big) \longrightarrow \mathcal{P}\big(\mathcal{N}(D)\big),$$

and $\mathcal{N}(D)$ is the set of normal edits in D, as defined in Definition 2.2.12.

In general we will assume that all categorical edit generation functions are normal unless specifically stated otherwise.

Instead of saying the long expression that "the covering sets of the failed edits of $\mathrm{FH}(E)$ are the error localisation solutions", we will say that the function FH is "covering set correctible", defined precisely by:

Definition 2.3.14. *The normal edit generation function G is **covering set correctible** if for each edit set E and for each record $\boldsymbol{v}$, we have that $\mathcal{CX}\big(G(E), \boldsymbol{v}\big) = \mathcal{EL}(E, \boldsymbol{v})$.*

Example 2.3.15. *In Example 2.3.10, we saw a demonstration of the covering set correctibility of* FH*, because $\mathcal{CX}\big(\mathrm{FH}(EE), \boldsymbol{vv}\big) = \mathcal{EL}(EE, \boldsymbol{vv})$.*

There are edit generation functions other than FH that are covering set correctible. In Sections 2.5 and 2.6, we will consider four such functions, which we call ENFH, MFH, MENFH and FCF_ω.

For convenience we will split covering set correctibility into two components, error correction guarantee and error correction totality, as in the next definition:

Definition 2.3.16. *The edit generation function G has the* ***error correction guarantee*** *if, for each edit set E and each record $\boldsymbol{v}$, we have that $\mathcal{CX}\big(G(E), \boldsymbol{v}\big) \subseteq \mathcal{EL}(E, \boldsymbol{v})$.*

The edit generation function G has ***error correction totality*** *if, for each edit set E and each record $\boldsymbol{v}$, we have that $\mathcal{CX}\big(G(E), \boldsymbol{v}\big) \supseteq \mathcal{EL}(E, \boldsymbol{v})$.*

In practice one seeks, not any covering set, but a "best" covering set, and so we define "smallest weighted covering set":

Definition 2.3.17. *A* ***smallest weighted covering set*** *C of the set E of normal edits, with a positive N-tuple $(b_1, \ldots, b_N)$ as weights, is a covering set of E such that:*

$$\text{if } C' \text{ is also a covering set of } E \text{ then } \textstyle\sum_{i \in C'} b_i \geq \sum_{i \in C} b_i \,.$$

As we did for covering sets, we will use a notation for smallest weighted covering sets:

Definition 2.3.18. *The set of smallest weighted covering sets of the set E, using the N-tuple of weights $(b_1, \ldots, b_N) = \boldsymbol{b}$, will be written $\mathcal{SC}(E, \boldsymbol{b})$.*

The set of the smallest weighted covering sets of the edits of E failed by $\boldsymbol{v}$, using the weights $(b_1, \ldots, b_N) = \boldsymbol{b}$, will be written $\mathcal{SCX}(E, \boldsymbol{v}, \boldsymbol{b})$. That is

$$\mathcal{SCX}(E, \boldsymbol{v}, \boldsymbol{b}) = \mathcal{SC}\big(\mathcal{X}(E, \boldsymbol{v}, \boldsymbol{b})\big).$$

Hence if $C \in \mathcal{SC}(E, \boldsymbol{b})$, where $\boldsymbol{b} = (b_1, \ldots, b_N)$, and $C' \in \mathcal{C}(E)$, then $\sum_{i \in C'} b_i \geq \sum_{i \in C} b_i$.

We would like the smallest weighted covering sets to be exactly the smallest weighted error localisation solutions. Hence we define the notion of "smallest covering set correctible", which is related to the notion of "covering set correctible".

Definition 2.3.19. *The edit generation function G is* ***smallest weighted covering set correctible (for the weights $\boldsymbol{b}$)*** *if for any edit set E and record $\boldsymbol{v}$, we have that $\mathcal{SCX}\big(G(E), \boldsymbol{v}, \boldsymbol{b}\big) = \mathcal{SEL}(E, \boldsymbol{v}, \boldsymbol{b})$, where $\mathcal{SEL}(E, \boldsymbol{v}, \boldsymbol{b})$ is the set of solutions to the smallest weighted error localisation problem for the record $\boldsymbol{v}$ with respect to the edit set E and weights vector $\boldsymbol{b}$, as defined in Definition 2.2.10.*

Smallest weighted covering set correctibility for all possible weights is equivalent to covering set correctibility. The proof is given below in Proposition 2.3.20.

However, if an edit generation function is covering set correctible for some but not all weights, then it need not be covering set correctible. We will give an example later, in Section 2.5, Example 2.5.16, after having defined the function FH and related functions.

Proposition 2.3.20. *The edit generation function G is covering set correctible if and only if, for each positive N-tuple $\boldsymbol{b}$ of weights, the function G is smallest weighted covering set correctible for $\boldsymbol{b}$.*

Proof.
Forward direction: immediate.
Backward direction: Suppose that, for each sequence $\boldsymbol{b}$ of N weights, the function G is smallest weighted covering set correctible for $\boldsymbol{b}$. That is, given a record $\boldsymbol{v}$, an edit set E, and a positive N-tuple $\boldsymbol{b}$, then $\mathcal{SCX}\big(G(E), \boldsymbol{v}, \boldsymbol{b}\big) = \mathcal{SEL}(E, \boldsymbol{v}, \boldsymbol{b})$. We will show that $\mathcal{CX}\big(G(E), \boldsymbol{v}\big) = \mathcal{EL}(E, \boldsymbol{v})$.

Firstly, we show that $\mathcal{CX}\big(G(E), \boldsymbol{v}\big) \subseteq \mathcal{EL}(E, \boldsymbol{v})$.

Let $C \in \mathcal{CX}\big(G(E), \boldsymbol{v}\big)$. We will show that $C \in \mathcal{EL}(E, \boldsymbol{v})$.

Define the N-tuple $\boldsymbol{d} = (d_1, \ldots, d_N)$ by:

$$d_i = \begin{cases} 1 & i \in C \\ N & i \notin C \end{cases}$$

Let $C' \in \mathcal{SCX}\big(G(E), \boldsymbol{v}, \boldsymbol{d}\big)$.

Then $C' \in \mathcal{SEL}(E, \boldsymbol{v}, \boldsymbol{d})$, since G is smallest weighted covering set correctible.

Then $C' \in \mathcal{EL}(E, \boldsymbol{v})$, by the definition of "smallest weighted covering set correctible" and "covering set correctible".

Also $C' \subseteq C$, because, if not

$$\begin{aligned} \sum_{i \in C'} d_i &= \big|C \cap C'\big| \;+\; N\big|C' \setminus C\big|, && \text{by the definition of } \boldsymbol{d} \\ &\geq N, && \text{since } C' \nsubseteq C \;\Rightarrow\; C' \setminus C \neq \emptyset \\ &> \big|C\big|, && \text{since } C' \setminus C \neq \emptyset \;\Rightarrow\; \big|C\big| < N \\ &= \sum_{i \in C} d_i\,, && \text{by the definition of } \boldsymbol{d}, \end{aligned}$$

which contradicts the fact that $C' \in \mathcal{SCX}\big(G(E), \boldsymbol{v}, \boldsymbol{d}\big)$.

Hence $C' \subseteq C$.

Hence $C \in \mathcal{EL}(E, \boldsymbol{v})$, since $C' \in \mathcal{EL}(E, \boldsymbol{v})$ and $C' \subseteq C$.

Hence $\mathcal{CX}\big(G(E), \boldsymbol{v}\big) \subseteq \mathcal{EL}(E, \boldsymbol{v})$.

Secondly, to show that $\mathcal{EL}(E, \boldsymbol{v}) \subseteq \mathcal{CX}\big(G(E), \boldsymbol{v}\big)$, use the same argument as above except with $\mathcal{EL}(E, \boldsymbol{v})$ swapped with $\mathcal{CX}\big(G(E), \boldsymbol{v}\big)$, and $\mathcal{SEL}(E, \boldsymbol{v}, \boldsymbol{d})$ swapped with $\mathcal{SCX}\big(G(E), \boldsymbol{v}, \boldsymbol{d}\big)$. ⊣

This section has outlined the covering set method and relevant properties for categorical edits. A similar method and properties apply to arithmetic edits expressed as linear inequalities. We leave the details to Chapter 7, but note here that the ideas were introduced by Fellegi and Holt (1976), and have been applied by Greenberg (1981), Greenberg and Surdi (1984), Winkler and Draper (1996), and Garcia and Goodwin (2002).

Before concluding this section, we summarise its main points. One method of error correction is to first do error localisation by the use of covering sets. For a given edit set E, a given record $\boldsymbol{v}$, and a chosen edit generation function G, we find the covering sets of the edits of $G(E)$ failed by $\boldsymbol{v}$ - that is, we find the covering sets $\mathcal{CX}\big(G(E), \boldsymbol{v}\big)$. If, for each E and $\boldsymbol{v}$, the covering sets $\mathcal{CX}\big(G(E), \boldsymbol{v}\big)$ are exactly the same as the error localisation solutions $\mathcal{EL}(E, \boldsymbol{v})$, then we say that G is covering set correctible. Indeed, we would prefer that G is smallest weighted covering set correctible: it turns out that smallest weighted covering set correctibility for all weights is equivalent to covering set correctibility. The challenge is to find functions G that are covering set correctible. One such function is FH, which we will define in the next section, where we also prove its covering set correctibility. Other such functions are ENFH, MFH, MENFH and FCF_ω, to be considered in the subsequent two sections.

2.4 The function FH

In this section, we define the edit generation function FH and prove that it is covering set correctible. In later sections, we will use FH to define other covering set correctible functions.

The function FH depends on another function, FHG, of edit sets.

Definition 2.4.1. *(Fellegi and Holt 1973, 1976) Let E be a set of normal edits, where each edit e in E is written $A_1^e \times \cdots \times A_N^e$. Then the* ***FH-generated edit*** *on E with* ***generating field*** *i is*

$$\mathrm{FHG}(i, E) = \prod_{j=1}^{i-1} \bigcap_{e \in E} A_j^e \quad \times \quad \bigcup_{e \in E} A_i^e \quad \times \prod_{j=i+1}^{N} \bigcap_{e \in E} A_j^e\,, \tag{2.4.1}$$

and the function $\mathrm{FHG} : \{1, \ldots, N\} \times \mathcal{P}\big(\mathcal{N}(D)\big) \longrightarrow \mathcal{P}(D)$*, where $\mathcal{N}(D)$ is defined in Definition 2.2.12 to be the set of normal edits in D.*

Note 1: The above expression is the normal form for $\mathrm{FHG}(i, E)$, except when $\mathrm{FHG}(i, E)$ is empty. When $\mathrm{FHG}(i, E)$ is empty, all components of its normal form are defined by Definition 2.2.12 to be empty, that is for all j, the set $A_j^{\mathrm{FHG}(i,E)} = \emptyset$, which is not necessarily the component calculated in the above expression.

Note 2: $\mathrm{FHG}(i, \emptyset) = \emptyset$.

Note 3: If e is a normal edit then $\mathrm{FHG}\big(i, \{e\}\big) = e$.

Example 2.4.2. *Suppose, as in the previous examples, that*

$$\begin{array}{rclcccl} D & = & A_{\mathsf{age}} & \times & A_{\mathsf{driver}} & \times & A_{\mathsf{grade}}\,, \\ e_1 & = & A_{\mathsf{age}} & \times & \{\mathrm{yes}\} & \times & \{1, 2, \ldots, 10\}, \\ e_2 & = & \{5, 6, \ldots, 9\} & \times & A_{\mathsf{driver}} & \times & \{7, 8, \ldots, 12\}. \end{array}$$

Then the edits $e_3, \ldots, e_8$ introduced in Examples 2.3.9 and 2.3.10 are constructed as follows:

$$\begin{array}{lclcccccc}
e_3 = \mathrm{FHG}\big(\mathsf{grade}, \{e_1, e_2\}\big) & = & \{5, 6, \ldots, 9\} & \times & \{\text{yes}\} & \times & A_{\mathsf{grade}}\,, \\
e_4 = \mathrm{FHG}\big(\mathsf{age}, \{e_1, e_2\}\big) & = & A_{\mathsf{age}} & \times & \{\text{yes}\} & \times & \{7, 8, 9, 10\}, \\
e_5 = \mathrm{FHG}\big(\mathsf{driver}, \{e_1, e_2\}\big) & = & \{5, 6, \ldots, 9\} & \times & A_{\mathsf{driver}} & \times & \{7, 8, 9, 10\}, \\
e_6 = \mathrm{FHG}\big(\mathsf{grade}, \{e_4, e_5\}\big) & = & \{5, 6, \ldots, 9\} & \times & \{\text{yes}\} & \times & \{7, 8, 9, 10\}, \\
e_7 = \mathrm{FHG}\big(\mathsf{driver}, \{e_1, e_3\}\big) & = & \{5, 6, \ldots, 9\} & \times & \{\text{yes}\} & \times & \{1, 2, \ldots, 10\}, \\
e_8 = \mathrm{FHG}\big(\mathsf{age}, \{e_2, e_3\}\big) & = & \{5, 6, \ldots, 9\} & \times & \{\text{yes}\} & \times & \{7, 8, \ldots, 12\}.
\end{array}$$

We now give the definition of Fellegi and Holt (1973, 1976) for the edit generation function $\mathrm{FH} : \mathcal{P}\big(\mathcal{N}(D)\big) \longrightarrow \mathcal{P}\big(\mathcal{N}(D)\big)$, where $\mathcal{N}(D)$ is defined in Definition 2.2.12 to be the set of normal edits in D. Given a normal edit set E, the edit set $\mathrm{FH}(E)$ is defined by the following process. A field i and a subset X of E are chosen such that $\mathrm{FHG}(i, X)$ is not already in E. The edit $\mathrm{FHG}(i, X)$ is then 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 set $\mathrm{FH}(E)$ is the edit set generated by the process.

A more formal inductive definition is:

Definition 2.4.3. *(Fellegi and Holt 1973, 1976) The* FH ***edit generation function***

$$\mathrm{FH} : \mathcal{P}\big(\mathcal{N}(D)\big) \longrightarrow \mathcal{P}\big(\mathcal{N}(D)\big)$$

is defined inductively for each set E of normal edits as follows:

1. $E \subseteq \mathrm{FH}(E)$*;*
2. *if $X \subseteq \mathrm{FH}(E)$ and i is a field, then* $\mathrm{FHG}(i, X) \in \mathrm{FH}(E)$.

Note: $\mathrm{FH}(\emptyset) = \{\emptyset\}$.

An equivalent definition of the function FH is in terms of sequences of normal edits, as stated in the next proposition.

Proposition 2.4.4. *Let E be a set of normal edits and let e be a normal edit. Then $e \in \mathrm{FH}(E)$ if and only if there exists a finite sequence of normal edits $e_1, \ldots, e_n = e$ such that for each $j = 1, \ldots, n$*

1. $e_j \in E$*, or*
2. *there is a subset X of $\{e_1, \ldots, e_{j-1}\}$ and a field i such that $e_j = \mathrm{FHG}(i, X)$.*

Proof. Apply Theorem 1.9.7 with:

1. $S =$ the set of normal edits $\mathcal{N}(D)$;
2. $B = E$; and
3. the relation R on $\mathcal{P}\big(\mathcal{N}(D)\big) \times \mathcal{N}(D)$ defined as follows: $(X, e) \in R$ if and only if there is a field i such that $e = \mathrm{FHG}(i, X)$. ⊣

We now turn to the proof that FH is covering set correctible. We will deal separately with the two components, error correction guarantee and error correction totality. We deal first with the more difficult component, the error correction guarantee, in Theorem 2.4.6 below, due to Fellegi and Holt (1973, 1976).

The main idea of the proof of the error correction guarantee of FH is the construction of a sequence of edit sets with a property called the "lifting property", which is the same as the "lifting principle" of de Waal (2003a, page 41).

Definition 2.4.5. *A sequence of edit sets* $(X_j)_{j=a}^{b}$ *, where* $0 \leq a < b \leq N$*, has the* ***lifting property for fields*** $i_a, \ldots, i_b$ *if the following holds:*

If $\boldsymbol{v}$ *is a record and* $a+1 \leq k \leq b$ *and* $\boldsymbol{v}$ *satisfies all edits in* X_k *then* $\{i_k\} \in \mathcal{EL}(X_{k-1}, \boldsymbol{v})$*.*

(That is, if $\boldsymbol{v}$ *satisfies all edits in* X_k *then the field* k *of* $\boldsymbol{v}$ *can be changed so that the new record satisfies all edits in* X_{k-1} *.)*

Theorem 2.4.6 (Error correction guarantee of FH**).** *The edit generation function* FH *has the error correction guarantee. (Fellegi and Holt (1973, 1976, Theorem 1, Corollaries 1, 2) and Liepins (1980b, Lemma 4, Corollaries 1, 2).)*

Outline of proof. For a given set E of explicit edits, Fellegi and Holt's proof depends on a sequence of edit sets $\Omega_N, \ldots, \Omega_0 = \mathrm{FH}(E)$, where each set Ω_k contains only those edits that are in $\mathrm{FH}(E)$ and in which the fields $1, \ldots, k$ are uninvolved[1]. That is

$$\Omega_k = \big\{ e \in \mathrm{FH}(E) \mid \text{fields } 1, \ldots, k \text{ are uninvolved in } e \big\}.$$

The proof is in three steps: the Lifting Property Theorem for $(\Omega_k)_{k=0}^{N}$ (F & H Theorem 1), the Repeated Lifting Corollary (F & H Corollary 1) and, finally, the Error Correction Guarantee Corollary (F & H Corollary 2). The details are below but I first give the ideas. The Lifting Property Theorem for $(\Omega_k)_{k=0}^{N}$ ensures that, so long as the record satisfies Ω_k for some k, it can be "improved" by changing the field k so as to satisfy Ω_{k-1}. Under the Repeated Lifting Corollary, if the record satisfies Ω_k for some k then it can be successively "improved" to satisfy $\Omega_k, \ldots, \Omega_0$, which is the set $\mathrm{FH}(E)$ of all edits generated by the Fellegi-Holt edit generation process. The proof of the Error Correction Guarantee Corollary finds an appropriate renumbering of the fields so that the record satisfies some Ω_k. Then, by the Repeated Lifting Corollary, the record can be successively improved to satisfy $\mathrm{FH}(E)$.

The proof of F & H Theorem 1 (Lifting Property Theorem) works by contradiction. First assume $(\Omega_k)_{k=0}^{N}$ does not have the lifting property. That is, assume that some record $\boldsymbol{v}$ satisfies all the edits in some Ω_k but $\{k\} \notin \mathcal{EL}(\Omega_{k-1}, \boldsymbol{v})$. This means that, no matter how the field k of $\boldsymbol{v}$ is changed, the new record will fail some edit in Ω_{k-1}. Hence, if the field k of record $\boldsymbol{v}$ is changed to the value a of field domain A_k, then the new record $\boldsymbol{v}^a$ will fail some edit e^a in Ω_{k-1}. The set $S = \{e^a \mid a \in A_k\}$ of such edits is

[1]Note that I have switched around the definition of Ω_k from that used by Fellegi and Holt, with a correspondingly changed result. Whereas I define the edits of Ω_k to have fields $1, \ldots, k$ uninvolved, Fellegi and Holt define the edits of Ω_k to have fields $k+1, \ldots, N$ uninvolved.

then used to generate a new edit $\gamma = \mathrm{FHG}(k, S)$. A calculation of the new edit γ shows that $\gamma \in \Omega_k$ and hence γ is satisfied by $\boldsymbol{v}$, since we assumed that $\boldsymbol{v}$ satisfies everything in Ω_k. Yet the calculation of γ also shows that γ is failed by $\boldsymbol{v}$. So $\{k\} \in \mathcal{EL}(\Omega_{k-1}, \boldsymbol{v})$, after all.

F & H Corollary 1 (Repeated Lifting Corollary) states that if $\boldsymbol{v}$ satisfies Ω_k then $\{1, \ldots, k\} \in \mathcal{EL}\big(\mathrm{FH}(E), \boldsymbol{v}\big)$. The proof is by repeated application of the Lifting Property Theorem, noting that $\Omega_0 = \mathrm{FH}(E)$.

F & H Corollary 2 (Error Correction Guarantee Corollary) states that FH has the error correction guarantee. To prove it, suppose that $C \in \mathcal{CX}\big(\mathrm{FH}(E), \boldsymbol{v}\big)$. Renumber the fields so that $C = \{1, \ldots, k\}$ where $k \geq 0$. Then $\boldsymbol{v}$ satisfies Ω_k and, by the Repeated Lifting Corollary, $C \in \mathcal{EL}\big(\mathrm{FH}(E), \boldsymbol{v}\big)$. Since $\mathrm{FH}(E) \supseteq E$, we have that $C \in \mathcal{EL}(E, \boldsymbol{v})$. ⊣

The error correction totality of FH is given below in Proposition 2.4.9. Although not explicitly proved by Fellegi and Holt, the error correction totality of FH does follow quickly from the following lemma of Fellegi and Holt, and its corollary.

Lemma 2.4.7. *(Fellegi and Holt 1973, 1976) Let E be a set of normal edits, i be a field, and $\boldsymbol{v}$ be a record that satisfies all edits in E. Then $\boldsymbol{v}$ satisfies the edit* $\mathrm{FHG}(i, E)$.

Outline of proof. Prove the contrapositive, as follows. Given an edit set E and a record $\boldsymbol{v}$, suppose that $\boldsymbol{v} \in \mathrm{FHG}(i, E)$. Then by the construction of $\mathrm{FHG}(i, E)$, the record $\boldsymbol{v}$ is in some edit e of E. ⊣

The next corollary directly extends the above lemma to the function FH. In the later logical formalisation of Chapter 3, the following property will be known as "soundness".

Corollary 2.4.8 ("Soundness" of FH**).** *Let E be a set of normal edits, and $\boldsymbol{v}$ be a record that satisfies all edits in E. Then $\boldsymbol{v}$ satisfies all edits in* $\mathrm{FH}(E)$.

We can now prove that FH has error correction totality.

Proposition 2.4.9 (Error correction totality of FH**).** *The edit generation function* FH *has error correction totality.*

Proof. Let E be an edit set and let $\boldsymbol{v} = (v_1, \ldots, v_N)$ be a record. We will show that $\mathcal{EL}(E, \boldsymbol{v}) \subseteq \mathcal{CX}\big(\mathrm{FH}(E), \boldsymbol{v}\big)$, by supposing that $C \in \mathcal{EL}(E, \boldsymbol{v})$ and showing that $C \in \mathcal{CX}\big(\mathrm{FH}(E), \boldsymbol{v}\big)$. We will do this by showing that for each edit e in $\mathcal{X}\big(\mathrm{FH}(E), \boldsymbol{v}\big)$ there is a field in C that is involved in e.

We first find a field k that is involved in e, as follows. Since $C \in \mathcal{EL}(E, \boldsymbol{v})$, then by Definition 2.2.8, there is a record $\boldsymbol{w} = (w_1, \ldots, w_N)$ that satisfies E and such that $C \supseteq \{j \mid w_j \neq v_j\}$. Then by Corollary 2.4.8, the record $\boldsymbol{w}$ satisfies all edits in $\mathrm{FH}(E)$, including the edit e, ie $\boldsymbol{w} \notin e$. Hence there is a field k such that $w_k \notin A_k^e$, so that $A_k^e \neq A_k$. Hence k is involved in e.

We now show that the field k is in C, as follows. Since $e \in \mathcal{X}\big(\mathrm{FH}(E), \boldsymbol{v}\big)$ we have that $\boldsymbol{v} \in e$, and hence $\boldsymbol{v}_k \in A_k^e$. Since $w_k \notin A_k^e$, we have that $w_k \neq v_k$, ie $k \in \{j \mid w_j \neq v_j\} \subseteq C$, from the properties of $\boldsymbol{w}$ given above.

But e is any edit in $\mathcal{X}\big(\mathrm{FH}(E), \boldsymbol{v}\big)$. Hence C covers $\mathcal{X}\big(\mathrm{FH}(E), \boldsymbol{v}\big)$, that is, $C \in \mathcal{CX}\big(\mathrm{FH}(E), \boldsymbol{v}\big)$, as required. ⊣

The proof that FH has error correction totality follows from its "soundness" (Corollary 2.4.8). Indeed, any "sound" edit generation function has error correction totality; the converse also applies: that is, "soundness" is equivalent to error correction totality. We will leave the proof to Chapter 5, after having formally defined soundness and error correction totality in terms of logic.

Before concluding this section, I present the following lemma, which we will need in later chapters. The lemma is about the function FHG applied to edits that are subsets of other edits.

Lemma 2.4.10. *(Generalisation of Kunnathur's Lemma 3.1 (Kunnathur 1982, page 38).) Suppose that E and E' are two edits sets; that $f : E \longrightarrow E'$ is surjective; and that if $e \in E$ then $e \subseteq f(e)$. Then for each field i, we have that* $\mathrm{FHG}(i, E) \subseteq \mathrm{FHG}(i, E')$.

Proof. Ignoring the order of the fields,

$$\begin{aligned}
\mathrm{FHG}(i, E) &= \prod_{j \neq i} \bigcap_{e \in E} A_j^e \times \bigcup_{e \in E} A_i^e \\
&\subseteq \prod_{j \neq i} \bigcap_{e \in E} A_j^{f(e)} \times \bigcup_{e \in E} A_i^{f(e)} && \text{since } e \subseteq f(e) \text{ and using Lemma 2.2.15} \\
&= \prod_{j \neq i} \bigcap_{e \in E'} A_j^e \times \bigcup_{e \in E'} A_i^e && \text{since } f \text{ is surjective} \\
&= \mathrm{FHG}(i, E').
\end{aligned}$$

⊣

This section has introduced the function FH and proved that it is covering set correctible. The proof of error correction totality is straightforward, but the proof of the error correction guarantee is more complex. There are other functions, derivable from FH, that are also covering set correctible, and they are the topics of the next two sections.

2.5 The functions ENFH, MFH and MENFH

This section introduces three more edit generation functions, ENFH, MFH, and MENFH, each of which is covering set correctible. Each of the three functions is a different subfunction of FH, where we defined subfunction in Chapter 1, Definition 1.9.3.

We start with the definitions of the three functions. Firstly, the function ENFH depends on a special type of FH-generated edit called an essentially new FH-generated edit:

Definition 2.5.1. *(based on Fellegi and Holt 1973, 1976.) Let i be a field, and let E be a set of normal edits, where each edit e in E is written $A_1^e \times \cdots \times A_N^e$. Then*

$\mathrm{FHG}(i, E)$ *is called an* ***essentially new FH-generated edit*** *if* $A_i^{\mathrm{FHG}(i,E)} = A_i$. *We also use the shorter term* ***essentially new edit****.*

Note 1: This means that each essentially new edit is non-empty.

Note 2: I have simplified Fellegi and Holt's definition of "essentially new implied edits", which included an additional property. The additional property was in terms of dominated edits, defined in Definition 2.2.5. In their definition of essentially new implied edits, Fellegi and Holt required, for each edit e of E, that no generating edits dominate the generated edit. That is, in Definition 2.5.1, they also required, for each edit e of E, that $A_i^e \subset A_i$. In this thesis, I do not impose this requirement on essentially new edits.

The function ENFH is defined inductively in the same way as FH, except that it depends on essentially new edits.

Definition 2.5.2. *The* ***essentially new FH edit generation function****, written* ENFH*, is the normal edit generation function*

$$\mathrm{ENFH} : \mathcal{P}\big(\mathcal{N}(D)\big) \longrightarrow \mathcal{P}\big(\mathcal{N}(D)\big)$$

defined inductively for each set E of normal edits by:

1. $E \subseteq \mathrm{ENFH}(E)$*;*

2. *if* $X \subseteq \mathrm{ENFH}(E)$*, and i is a field, and* $\mathrm{FHG}(i, X)$ *is essentially new, then* $\mathrm{FHG}(i, X) \in \mathrm{ENFH}(E)$*.*

Example 2.5.3. *Suppose, as in the previous examples, that*

$$\begin{array}{rcccccc} D & = & A_{\mathsf{age}} & \times & A_{\mathsf{driver}} & \times & A_{\mathsf{grade}}\,, \\ EE & = & \{e_1, e_2\}, \text{ where} & & & & \\ e_1 & = & A_{\mathsf{age}} & \times & \{\text{yes}\} & \times & \{1, 2, \ldots, 10\}, \text{ and} \\ e_2 & = & \{5, 6, \ldots, 9\} & \times & A_{\mathsf{driver}} & \times & \{7, 8, \ldots, 12\}. \end{array}$$

Then $\mathrm{ENFH}(EE) = \{e_1, e_2, e_3, e_4, e_5\}$*,where e_3, e_4 and e_5 were constructed in Example 2.4.2 as:*

$$\begin{array}{rcccccc} e_3 = \mathrm{FHG}(\mathsf{grade}, EE) & = & \{5, 6, \ldots, 9\} & \times & \{\text{yes}\} & \times & A_{\mathsf{grade}}\,, \\ e_4 = \mathrm{FHG}(\mathsf{age}, EE) & = & A_{\mathsf{age}} & \times & \{\text{yes}\} & \times & \{7, 8, 9, 10\}, \\ e_5 = \mathrm{FHG}(\mathsf{driver}, EE) & = & \{5, 6, \ldots, 9\} & \times & A_{\mathsf{driver}} & \times & \{7, 8, 9, 10\}. \end{array}$$

The other two functions to be introduced in this section are MFH and MENFH, defined below. The two functions depend on the idea, introduced by Liepins (1980b, Corollary 2, page 15), of considering edit generation functions that return only maximal edits. The two functions are defined as follows.

Definition 2.5.4. *The* ***maximal FH edit generation function*** *, written* MFH*, is the normal edit generation function defined as follows. Given an edit set E,*

$$\mathrm{MFH}(E) = \mathrm{Max}\big(\mathrm{FH}(E)\big),$$

*where "*Max*" means taking the maximal elements with respect to subset inclusion, as in Definition 1.9.2.*

Definition 2.5.5. *The* ***maximal essentially new FH edit generation function****, written* MENFH*, is the normal edit generation function defined as follows. Given an edit set E,*

$$\mathrm{MENFH}(E) = \mathrm{Max}\big(\mathrm{ENFH}(E)\big),$$

*where "*Max*" means taking the maximal elements with respect to subset inclusion, as in Definition 1.9.2.*

Note: Fellegi and Holt had already excluded some dominated edits in their definition of "essentially new", but did not restrict the function to just maximal edits. Their definition of "essentially new" prevents a generated essentially new edit from being dominated by any of its generating edits.

Example 2.5.6. *As in the previous examples, suppose that*

$$\begin{array}{rcccccl}
D & = & A_{\mathsf{age}} & \times & A_{\mathsf{driver}} & \times & A_{\mathsf{grade}}\,, \\
EE & = & \{e_1, e_2\}, \textit{ where} & & & & \\
e_1 & = & A_{\mathsf{age}} & \times & \{\mathrm{yes}\} & \times & \{1, 2, \ldots, 10\}, \quad \textit{and} \\
e_2 & = & \{5, 6, \ldots, 9\} & \times & A_{\mathsf{driver}} & \times & \{7, 8, \ldots, 12\}.
\end{array}$$

As given in Example 2.3.10, we have that $\mathrm{FH}(EE) = \{e_1, \ldots, e_8\}$*, where*

$$\begin{array}{rcccccl}
e_3 & = & \{5, 6, \ldots, 9\} & \times & \{\mathrm{yes}\} & \times & A_{\mathsf{grade}}\,, \\
e_4 & = & A_{\mathsf{age}} & \times & \{\mathrm{yes}\} & \times & \{7, 8, 9, 10\}, \\
e_5 & = & \{5, 6, \ldots, 9\} & \times & A_{\mathsf{driver}} & \times & \{7, 8, 9, 10\}, \\
e_6 & = & \{5, 6, \ldots, 9\} & \times & \{\mathrm{yes}\} & \times & \{7, 8, 9, 10\}, \\
e_7 & = & \{5, 6, \ldots, 9\} & \times & \{\mathrm{yes}\} & \times & \{1, 2, \ldots, 10\}, \\
e_8 & = & \{5, 6, \ldots, 9\} & \times & \{\mathrm{yes}\} & \times & \{7, 8, \ldots, 12\}.
\end{array}$$

We then have that $\mathrm{MFH}(EE) = \mathrm{MENFH}(EE) = \{e_1, e_2, e_3\}$.

Having defined the functions ENFH, MFH and MENFH, we now consider their covering set correctibility. We first consider their error correction totality, which follows from the next lemma and corollary. The lemma is about the connections between covering sets, the function $\mathcal{X}$, and subsets of edit sets; while its corollary is about the connection between subfunctions and error correction totality.

Lemma 2.5.7. *Let E, E_1 and E_2 be sets of normal clauses and let $\boldsymbol{v}$ be a record. Then the following hold:*

1. *If* $E_1 \subseteq E_2$ *, then* $\mathcal{C}(E_1) \supseteq \mathcal{C}(E_2)$.
2. *If* $E_1 \subseteq E_2$ *, then* $\mathcal{X}(E_1, \boldsymbol{v}) \subseteq \mathcal{X}(E_2, \boldsymbol{v})$.
3. *If* H *is a subfunction of the edit generation function* G*, then* $\mathcal{CX}\big(H(E), \boldsymbol{v}\big) \supseteq \mathcal{CX}\big(G(E), \boldsymbol{v}\big)$.

Proof.

1. Let $C \in \mathcal{C}(E_2)$, and let $e \in E_1$. Since $E_1 \subseteq E_2$, we have that $e \in E_2$, and hence there is a field i in C that is involved in e. Hence C covers E_1 , as required.
2. Let $e \in \mathcal{X}(E_1, \boldsymbol{v})$. Then $e \in E_1$ and hence $e \in E_2$. Also $\boldsymbol{v}$ fails the edit e. Hence $e \in \mathcal{X}(E_2, \boldsymbol{v})$.
3. Since H is a subfunction of G, we have that $H(E) \subseteq G(E)$. Then by part 2 of this lemma, $\mathcal{X}\big(H(E), \boldsymbol{v}\big) \subseteq \mathcal{X}\big(G(E), \boldsymbol{v}\big)$, and hence the result follows from part 1 of this lemma. ⊣

Corollary 2.5.8. *Each subfunction of an edit generation function with error correction totality also has error correction totality.*

Proof. Let G be an edit generation function with error correction totality. Let H be a subfunction of G. Let E be an edit set and let $\boldsymbol{v}$ be a record. Since H is a subfunction of G, then by Lemma 2.5.7 we have that $\mathcal{CX}\big(H(E), \boldsymbol{v}\big) \supseteq \mathcal{CX}\big(G(E), \boldsymbol{v}\big)$. But since G has error correction totality, we have that $\mathcal{CX}\big(G(E), \boldsymbol{v}\big) \supseteq \mathcal{EL}(E, \boldsymbol{v})$, and thus $\mathcal{CX}\big(H(E), \boldsymbol{v}\big) \supseteq \mathcal{EL}(E, \boldsymbol{v})$. Hence H has error correction totality. ⊣

Corollary 2.5.9. *The functions* ENFH*,* MFH *and* MENFH *have error correction totality.*

Proof. The functions ENFH, MFH and MENFH are subfunctions of FH, which has error correction totality (by Proposition 2.4.9). ⊣

Having dealt with error correction totality, we are now ready to consider the second component of covering set correctibility, namely the error correction guarantee.

But first, we note that Corollary 2.5.8, about subfunctions and error correction totality, has a dual, presented below as Corollary 2.5.10, about superfunctions and error correction guarantee. Although not relevant in this chapter, the corollary will be relevant in Chapter 6.

Corollary 2.5.10. *Each superfunction of an edit generation function with the error correction guarantee also has the error correction guarantee.*

Proof. Let H be an edit generation function with the error correction guarantee. Let G be a superfunction of H. Let E be an edit set and let $\boldsymbol{v}$ be a record. Since H is a subfunction of G, then by Lemma 2.5.7 we have that $\mathcal{CX}\big(H(E), \boldsymbol{v}\big) \supseteq \mathcal{CX}\big(G(E), \boldsymbol{v}\big)$. But since H has the error correction guarantee, we have that $\mathcal{EL}(E, \boldsymbol{v}) \supseteq \mathcal{CX}\big(H(E), \boldsymbol{v}\big)$, and thus $\mathcal{EL}(E, \boldsymbol{v}) \supseteq \mathcal{CX}\big(G(E), \boldsymbol{v}\big)$. Hence G has the error correction guarantee. ⊣

We now turn to the error correction guarantee of the three functions ENFH, MFH and MENFH. We first consider the function ENFH. Although Fellegi and Holt did not explicitly prove that ENFH has the error correction guarantee, they did imply its proof because of its similarity with the proof of the error correction guarantee of FH.

Proposition 2.5.11. ENFH *has the error correction guarantee.*

Proof. Exactly the same proof as for FH, noting that all edits generated in the proof are essentially new. ⊣

We now consider the error correction guarantee of the two functions MFH and MENFH. Their error correction guarantee follows from the next lemma and corollary, which are duals of the above Lemma 2.5.7 and Corollary 2.5.8. The lemma is about the connections between covering sets, the function $\mathcal{X}$, and superior sets (Definition 1.9.4 of Chapter 1); while its corollary is about superior functions (also defined in Definition 1.9.4).

Lemma 2.5.12. *Let E, E_1 and E_2 be sets of normal clauses and let $\boldsymbol{v}$ be a record. Then the following hold:*

1. *If E_1 is superior to E_2, then $\mathcal{C}(E_1) \subseteq \mathcal{C}(E_2)$.*
2. *If E_1 is superior to E_2, then $\mathcal{X}(E_1, \boldsymbol{v})$ is superior to $\mathcal{X}(E_2, \boldsymbol{v})$.*
3. *If H is a superior function to the edit generation function G, then $\mathcal{CX}\big(H(E), \boldsymbol{v}\big) \subseteq \mathcal{CX}\big(G(E), \boldsymbol{v}\big)$.*

Proof.

1. Let $C \in \mathcal{C}(E_1)$, and let $e_2 \in E_2$. Since E_1 is superior to E_2, there is an edit e_1 in E_1 such that $e_1 \supseteq e_2$. Since $C \in \mathcal{C}(E_1)$, there is a field i in C that is involved in e_1, ie $A_i^{e_1} \neq A_i$. Since $e_1 \supseteq e_2$, we have that $A_i^{e_1} \supseteq A_i^{e_2}$ and so $A_i^{e_2} \neq A_i$. That is, the field i is involved in e_2. Hence $C \in \mathcal{C}(E_2)$.
2. Let $e_2 \in \mathcal{X}(E_2, \boldsymbol{v})$. Then $e_2 \in E_2$, and since E_1 is superior to E_2, there is an edit e_1 in E_1 such that $e_1 \supseteq e_2$. Also, since $e_2 \in \mathcal{X}(E_2, \boldsymbol{v})$, we have that $\boldsymbol{v} \in e_2$, and hence also $\boldsymbol{v} \in e_1$. Hence $e_1 \in \mathcal{X}(E_1, \boldsymbol{v})$. Hence $\mathcal{X}(E_1, \boldsymbol{v})$ is superior to $\mathcal{X}(E_2, \boldsymbol{v})$, as required.
3. Since H is a superior function to G, we have that $H(E)$ is superior to $G(E)$. Then by part 2 of this lemma, $\mathcal{X}\big(H(E), \boldsymbol{v}\big)$ is superior to $\mathcal{X}\big(G(E), \boldsymbol{v}\big)$, and hence the result follows from part 1 of this lemma.

⊣

Corollary 2.5.13. *If the edit generation function G has the error correction guarantee, and H is a superior function to G, then H has the error correction guarantee.*

Proof. Let $\boldsymbol{v}$ be a record, and let E be an edit set. Since the set $H(E)$ is superior to the set $G(E)$, then by Lemma 2.5.12 we have that $\mathcal{CX}\big(H(E), \boldsymbol{v}\big) \subseteq \mathcal{CX}\big(G(E), \boldsymbol{v}\big)$. But since G has the error correction guarantee we have that $\mathcal{CX}\big(G(E), \boldsymbol{v}\big) \subseteq \mathcal{EL}(E, \boldsymbol{v})$. Hence $\mathcal{CX}\big(H(E), \boldsymbol{v}\big) \subseteq \mathcal{EL}(E, \boldsymbol{v})$, and thus H has the error correction guarantee. ⊣

Corollary 2.5.14. *The functions* MFH *and* MENFH *have the error correction guarantee.*

Proof. By Lemma 1.9.5 part 2, the functions MFH and MENFH are superior functions to FH and ENFH respectively, and both FH and ENFH have the error correction guarantee. ⊣

Note: Kunnathur (1982, Lemma 3.1, page 38) gave the essence of the proof that dominated edits can be removed in the edit generation process without affecting the error correction guarantee.

We have now shown that the three functions ENFH, MFH and MENFH have both error correction totality and the error correction guarantee. Therefore we have completed the main purpose of this section, which is to show that the three functions are covering set correctible.

Before concluding this section, I present two more items. The first item is to note that Corollary 2.5.13, about superior functions and error correction guarantee, has a dual, presented below as Corollary 2.5.15, about error correction totality. Although not relevant in this chapter, the corollary will be relevant in Chapter 6.

Corollary 2.5.15. *If the edit generation function H has error correction totality, and H is a superior function to G, then G has error correction totality.*

Proof. Let $\boldsymbol{v}$ be a record, and let E be an edit set. Since the set $H(E)$ is superior to the set $G(E)$, then by Lemma 2.5.12 we have that $\mathcal{CX}\big(H(E), \boldsymbol{v}\big) \subseteq \mathcal{CX}\big(G(E), \boldsymbol{v}\big)$. But since H has error correction totality we have that $\mathcal{EL}(E, \boldsymbol{v}) \subseteq \mathcal{CX}\big(H(E), \boldsymbol{v}\big)$. Hence $\mathcal{EL}(E, \boldsymbol{v}) \subseteq \mathcal{CX}\big(G(E), \boldsymbol{v}\big)$, and thus G has error correction totality. ⊣

The second item that I present before concluding this section is the example foreshadowed in Section 2.3, page 27, which was deferred until now because it depends on the function ENFH. The example demonstrates that a function that is smallest weighted covering set correctible for a given set of weights need not be covering set correctible.

Example 2.5.16. *Suppose that $N = 2$, that $A_1 = \{1, 2, 3, 4\}$, that $A_2 = \{1, 2\}$, that $\boldsymbol{b} = (1, 2)$, that the edit set $X = \big\{\{2, 3\} \times \{1\}, \{3, 4\} \times \{2\}\big\}$, and that $G(E)$ is defined for each edit set E by*

$$G(E) = \begin{cases} \text{ENFH}(E), & \text{if } E \neq X \\ E, & \text{if } E = X. \end{cases}$$

We will show that G is smallest weighted covering set correctible for the weights $(1, 2)$ but is not covering set correctible.

Firstly, to confirm that G is smallest weighted covering set correctible for the weights $(1,2)$, we show, for each edit set E and each record $\boldsymbol{v}$, that

$$\mathcal{SCX}\big(G(E),\boldsymbol{v},(1,2)\big)=\mathcal{SEL}\big(E,\boldsymbol{v},(1,2)\big),$$

by considering three cases:

Case 1: $E\neq X$. We can use the fact that ENFH *is covering set correctible, and hence is smallest weighted covering set correctible for any set of weights:*

$$\begin{aligned}\mathcal{SCX}\big(G(E),\boldsymbol{v},(1,2)\big)&=\mathcal{SCX}\big(\mathrm{ENFH}(E),\boldsymbol{v},(1,2)\big)\,,\ \textit{by the definition of } G\\&=\mathcal{SEL}\big(E,\boldsymbol{v},(1,2)\big),\ \textit{because ENFH is smallest weighted covering set correctible for any set of weights.}\end{aligned}$$

Case 2: $E=X$ and $\boldsymbol{v}$ satisfies X. Then

$$\begin{aligned}\mathcal{SCX}\big(G(E),\boldsymbol{v},(1,2)\big)&=\mathcal{SCX}\big(E,\boldsymbol{v},(1,2)\big),\ \textit{by the definition of } G\\&=\mathcal{SC}(\emptyset),\ \textit{since } \boldsymbol{v} \textit{ satisfies } E\\&=\{\emptyset\}\\&=\mathcal{SEL}\big(E,\boldsymbol{v},(1,2)\big),\ \textit{since } \boldsymbol{v} \textit{ satisfies } E.\end{aligned}$$

Case 3: $E=X$ and $\boldsymbol{v}$ fails X. Then

$$\begin{aligned}\mathcal{SCX}\big(G(E),\boldsymbol{v},(1,2)\big)&=\mathcal{SCX}\big(E,\boldsymbol{v},(1,2)\big),\ \textit{by the definition of } G\\&=\{\textit{field } 1\},\ \textit{since field } 1 \textit{ is involved in both edits of } X\textit{, and } \{\textit{field } 1\} \textit{ is the smallest weighted non-empty set of fields}\\&=\mathcal{SEL}\big(E,\boldsymbol{v},(1,2)\big),\ \textit{since any record whose first field equals } 1 \textit{ satisfies all of } E \textit{ and } \{\textit{field } 1\} \textit{ is the smallest weighted non-empty set of fields.}\end{aligned}$$

Having confirmed that G is smallest weighted covering set correctible for the weights $(1,2)$, we now confirm that G is not covering set correctible. Consider the record $\boldsymbol{v}=(3,1)$, which fails the first edit in the set X and satisfies the second. Then $\{\textit{field } 2\}\in\mathcal{CX}\big(G(X),\boldsymbol{v}\big)$, because $G(X)=X$ and field 2 is uninvolved in the first edit of X. But $\{\textit{field } 2\}\notin\mathcal{EL}(X,\boldsymbol{v})$, because the only possible correction using only field 2, namely $(3,2)$, fails the second edit of X. Hence $\mathcal{CX}\big(G(X),\boldsymbol{v}\big)\neq\mathcal{EL}(X,\boldsymbol{v})$ and so G is not covering set correctible.

Thus G is smallest weighted covering set correctible for $(1,2)$, but is not covering set correctible.

This section has introduced the three functions ENFH, MFH and MENFH, which are subfunctions of FH. They are all covering set correctible. One direction of their covering set correctibility, namely their error correction totality, follows from their being

subfunctions of FH. As for the other direction, namely the error correction guarantee, the proof for ENFH is different from that for MFH and MENFH. The proof of error correction guarantee of ENFH is essentially the same as that for FH, while the error correction guarantee of MFH and MENFH follows from their being superior functions of FH and ENFH respectively.

There is one more covering set correctible function to be considered: it is the function FCF_ω. Its definition is more complex, and is left to the next section.

2.6 The Field Code Forest Algorithm

In this section, we define the edit generation function FCF_ω, or rather, the set of such functions: there is one function of form FCF_ω for each permutation ω of the fields. As to the covering set correctibility of the functions FCF_ω, its proof is significantly more complex than the other proofs of covering set correctibility and is left to Chapter 6.

The letters FCF stand for "Field Code Forest", because the function FCF_ω is specified by the so-called Field Code Forest Algorithm, abbreviated as FCF Algorithm. The FCF Algorithm was developed by Liepins, Kunnathur and Garfinkel (Liepins 1980a, 1984; Kunnathur 1982; Garfinkel, Kunnathur, and Liepins 1984, 1986b).

There have been questions about whether the FCF Algorithm is correct, in the sense of whether the corresponding function FCF_ω has covering set correctibility (Winkler 1997). Although Garfinkel et al. (1986b) presented a proof of what they called the "sufficiency" of the FCF Algorithm, we will see in Chapter 6 that their proof proves something other than the correctness of the FCF Algorithm.

Each function FCF_ω is a subfunction of ENFH. The algorithm gives a systematic way of choosing generating fields, and considerably reduces the size of the search space of generating fields.

The FCF Algorithm finds implied edits as it traverses a field code forest, which is in fact a tree rather than a forest. Its nodes are labelled with subsets of the set $\{1, \ldots, N\}$ such that each subset of $\{1, \ldots, N\}$ labels exactly one node of the tree. An example, for $N = 4$, is given in Figure 2.1, where each subset of $\{1, \ldots, N\}$ is represented by a corresponding string.

The more detailed definition of a field code forest is:

Definition 2.6.1. *A **field code forest F(ω)** is defined for each set $\{1, \ldots, N\}$ and each total ordering (or permutation) $\omega = (o_1, \ldots, o_N)$ of $\{1, \ldots, N\}$. Here the o_k's are distinct elements of $\{1, \ldots, N\}$. The Field Code Forest is a labelled tree, labelled by strings over the alphabet $\{1, \ldots, N\}$.*

1. *The root node is labelled with the empty string ().*
2. *If a node* n *is labelled with the string S then:*
   - *if $o_N \in S$ then* n *has no children, and*
   - *if $o_N \notin S$ and $K = \mathrm{Max}\big(\{0\} \cup \{k \mid o_k \in S\}\big)$ then* n *has $N - K$ children, labelled with $S \circ (o_j)$ for $j \in \{K+1, \ldots, N\}$, where $\circ$ means concatenation of strings. (Note that $K < N$.)*

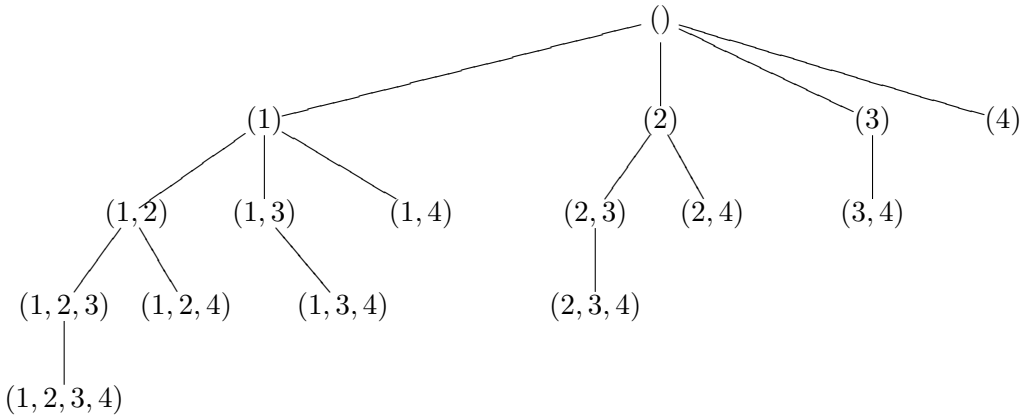

Figure 2.1: The field code forest $\mathrm{F}\big((1,2,3,4)\big)$.

We will use the same symbol to represent both the node's string and the corresponding subset, whenever there is no ambiguity.

Some simple consequences of the definition are:

- Each subset of $\{1,\ldots,N\}$ appears exactly once in the field code forest.
- Hence the node's label (string or subset) can be used to name the node.
- The order in which fields appear in each label is the same as their order in ω.
- There are N child nodes of the root node, labelled with $(o_1),\ldots,(o_N)$ respectively.
- The order of the nodes in the tree depends on the ordering of ω.

Liepins, Kunnathur and Garfinkel give a slightly different definition of a field code forest: they omit the root node () and thereby have a set of trees, hence the use of the word "forest". For our purposes the two definitions are equivalent; our definition makes the explanation of the algorithm slightly shorter.

The FCF Algorithm works through the forest in a depth-first order.[2] A precise statement of the Algorithm is in later paragraphs but in words it is described as follows. We start with a set E of explicit edits. At each node σ we generate an edit set associated with node σ and called $\mathrm{GenI}(\sigma,E)$ (where "I" stands for "initial"). The edit set associated with the node σ may be modified as the algorithm traverses the nodes: the edit set at node σ just after the node τ has been traversed will be called $\mathrm{GenV}(\sigma,E,\tau)$ (where "V" stands for "varying"). After the last node of the field code forest has been traversed the edit set associated with the node σ will be called $\mathrm{GenF}(\sigma,E)$ (where "F" stands for "final"). Hence, if ν is the last node of the field code forest, then $\mathrm{GenF}(\sigma,E)=\mathrm{GenV}(\sigma,E,\nu)$.

[2]In fact any order works so long as deeper nodes are considered after higher nodes on the same branch.

The set $\mathrm{GenI}(\sigma, E)$ is calculated using FH edit generation where the generating field is the new field added at the node σ. The edits that may be used at the node labelled $\sigma = (i_1, \ldots, i_k)$ are not all previously generated edits: rather they are only the edits generated in nodes in the branch above the current node with fields $i_1, \ldots, i_{k-1}$ uninvolved and with field i_k involved. Only certain edits are allowed at node $(i_1, \ldots, i_k)$: in any generated edit the field i_k must be uninvolved. If no edits can be generated at a node then the remainder of that branch is not traversed.

After calculating the set $\mathrm{GenI}(\sigma, E)$, each node τ traversed thus far is revisited. At each node τ the current edit set is updated by replacing any dominated edits by the maximal generated edits that dominate them. This gives the set $\mathrm{GenV}(\tau, E, \sigma)$.

A precise statement of the FCF Algorithm follows. By way of an example, Tables 2.1 and 2.2, in the appendix to this chapter, give the calculation for the explicit edit set E defined in Example 2.9.1 on page 50.

Definition 2.6.2 (Statement of the FCF Algorithm).

INPUT TO THE ALGORITHM: *a set E of edits on fields $\{1, \ldots, N\}$, and a total ordering ω of the set $\{1, \ldots, N\}$.*

STEPS IN THE ALGORITHM: *Traverse the field code forest $F(\omega)$ in depth-first order. At each node σ of $F(\omega)$:*

(i) Calculate the set $\mathrm{GenI}(\sigma, E)$ according to Definition 2.6.4 below.

(ii) Calculate the set $\mathrm{GenV}(\sigma, E, \sigma)$ using Domination Rule 1 given in Definition 2.6.5 below.

(iii) For each node τ visited before σ, calculate $\mathrm{GenV}(\tau, E, \sigma)$ using Domination Rule 2 given in Definition 2.6.5 below.

For each node σ, let $\mathrm{GenF}(\sigma, E)$ be the set of edits at node σ after the last node ν of the field code forest has been traversed. That is, $\mathrm{GenF}(\sigma, E) = \mathrm{GenV}(\sigma, E, \nu)$.

OUTPUT OF THE ALGORITHM: *the set*

$$\mathrm{FCF}_\omega(E) = \bigcup \big\{ \mathrm{GenF}(\sigma, E) \;\big|\; \sigma \text{ is a node of } F(\omega) \big\}.$$

In order to define $\mathrm{GenI}(\sigma, E)$, we define the set $\mathrm{BranchV}(\sigma, E, \tau)$ to be the set of edits appearing in the tree at or above node σ just after node τ has been traversed, as follows:

Definition 2.6.3. *Suppose E is an edit set and ω is a total ordering of the fields. Suppose also that σ and τ are nodes of the field code forest $F(\omega)$, and that either $\sigma = \tau$ or τ is traversed after σ. Write $\sigma = (i_1, \ldots, i_k)$ where $k \geq 0$. Then*

$$\mathrm{BranchV}(\sigma, E, \tau) = \bigcup \big\{ \mathrm{GenV}\big((i_1, \ldots, i_j), E, \tau\big) \;\big|\; j = 0, \ldots, k \big\}.$$

We can now define the set $\mathrm{GenI}(\sigma, E)$.

Definition 2.6.4 (Definition of $\mathrm{GenI}(\sigma, E)$ for step (i) of Definition 2.6.2). *Let $\sigma = (i_1, \ldots, i_m) \subseteq \omega$. Define the set $\mathrm{GenI}\big((i_1, \ldots, i_m), E\big)$ inductively as follows:*

a. $\mathrm{GenI}\big((), E\big) = E$.

b. If $m \geq 1$ and $\mathrm{GenV}\big((i_1, \ldots, i_{m-1}), E, (i_1, \ldots, i_{m-1})\big) = \emptyset$, then

$$\mathrm{GenI}\big((i_1, \ldots, i_m), E\big) = \emptyset.$$

c. If $m \geq 1$ and $\mathrm{GenV}\big((i_1, \ldots, i_{m-1}), E, (i_1, \ldots, i_{m-1})\big) \neq \emptyset$ then

$$\begin{aligned}\mathrm{GenI}\big((i_1, \ldots, i_m), E\big) = \Big\{ & \mathrm{FHG}(i_m, X) \mid \\ & X \subseteq \mathrm{BranchV}\big((i_1, \ldots, i_{m-1}), E, (i_1, \ldots, i_{m-1})\big), \text{ and} \\ & \mathrm{FHG}(i_m, X) \neq \emptyset, \text{ and} \\ & i_m \text{ is not involved in } \mathrm{FHG}(i_m, X), \text{ and} \\ & i_m \text{ is involved in each element of } X, \text{ and} \\ & \text{each of } i_1, \ldots, i_{m-1} \text{ is uninvolved in each element of } X \Big\}.\end{aligned}$$

For the purposes of later proofs, it is useful to list and number the properties in part c of Definition 2.6.4 of GenI given above, as follows:

For $m \geq 1$, the edit $\alpha \in \mathrm{GenI}\big((i_1, \ldots, i_m), E\big)$ if and only if

G1. $\mathrm{GenI}\big((i_1, \ldots, i_{m-1}), E\big) \neq \emptyset$;

G2. $\alpha = \mathrm{FHG}(i_m, X)$;

G3. $X \subseteq \mathrm{BranchV}\big((i_1, \ldots, i_{m-1}), E, (i_1, \ldots, i_{m-1})\big)$;

G4. $\alpha \neq \emptyset$;

G5. the field i_m is not involved in α;

G6. the field i_m is involved in each element of X;

G7. each of the fields $i_1, \ldots, i_{m-1}$ is uninvolved in each element of X.

The statement of the FCF Algorithm of Definition 2.6.2 also depends on the following domination rules.

Definition 2.6.5 (Domination rules for steps (ii) and (iii) of Definition 2.6.2). *Let S_σ be the set of nodes visited prior to visiting σ. If $\sigma \neq ()$, then let σ' be the node visited immediately before σ. The domination rules, applied after calculating $\mathrm{GenI}(\sigma, E)$, are:*

Domination Rule 1. Calculate $\mathrm{GenV}(\sigma, E, \sigma)$:

(a) *If* $\sigma = ()$, *then remove all dominated edits from* $\mathrm{GenI}(\sigma, E)$. *That is,*

$$\mathrm{GenV}\big(((), E, ()\big) = \mathrm{Max} \circ \mathrm{GenI}\big(((), E\big).$$

(b) *If* $\sigma \neq ()$, *then replace each edit in* $\mathrm{GenI}(\sigma, E)$ *by all maximal dominating edits already generated. That is,*

$$\mathrm{GenV}(\sigma, E, \sigma) = \mathrm{Max}\Big(\mathrm{GenI}(\sigma, E) \cup \big\{\beta \mid \beta \in \bigcup_{\tau \in S_\sigma} \mathrm{GenV}(\tau, E, \sigma'),\ \textit{and there is an } \alpha \textit{ in } \mathrm{GenI}(\sigma, E) \textit{ such that } \beta \supset \alpha\big\}\Big).$$

Domination Rule 2. If $S_\sigma \neq \emptyset$, *then for each* τ *in* S_σ, *calculate* $\mathrm{GenV}(\tau, E, \sigma)$ *as follows: Replace any already generated edit by all maximal edits that contain it and that are in* $\mathrm{GenV}(\sigma, E, \sigma)$. *That is,*

$$\mathrm{GenV}(\tau, E, \sigma) = \mathrm{Max}\Big(\mathrm{GenV}(\tau, E, \sigma') \cup \big\{\beta \mid \beta \in \mathrm{GenV}(\sigma, E, \sigma),\ \textit{and there is an } \alpha \textit{ in } \mathrm{GenV}(\tau, E, \sigma') \textit{ such that } \beta \supset \alpha\big\}\Big).$$

The above algorithm statement differs in places from those given in the five papers by Garfinkel, Kunnathur and Liepins (G, K & L 1984, 1986b; Kunnathur 1982; Liepins 1980a, 1984), which differ in places from each other. The appendix to this chapter gives details.

Chapter 6 will introduce a modified version of the FCF Algorithm, called the FCFS Algorithm, which is the same as the FCF Algorithm, except without the Domination Rules. We will use the FCFS Algorithm in the proof of the covering set correctibility of the function FCF_ω.

In this section we have defined the function FCF_ω in terms of the FCF Algorithm. Although the definition of the function is complex, it is a subfunction of the more easily defined function ENFH. The proof of its covering set correctibility is also complex, and we leave it to Chapter 6.

2.7 Practicalities of the covering set method

The covering set method is slow and uses much memory, because the number of generated edits increases exponentially with the number of explicit edits. As a consequence, several improvements have been suggested to the edit generation process. This section describes some of the methods to reduce the number of edits generated.

The first method for reducing the number of generated edits is to seek only maximal edits, as in the functions MFH, MENFH and FCF_ω. An additional improvement is to prevent the generation of dominated edits rather than merely deleting them after generating them. Some examples, further discussed by Winkler (1997, 1998), are as follows.

1. The edit generation step $e = \mathrm{FHG}(i, X)$ need only be calculated if the generating field i is involved in each edit of the generating set X. Otherwise the generated edit e is dominated by one of the edits of X. (Fellegi and Holt 1976, page 29, item 1.)

2. The edit generation step $e = \mathrm{FHG}(i, X)$ need not be calculated if there is an edit e' in the edit set X, a superset X' of $X \setminus \{e'\}$, and a field k (which may or may not equal i) such that $e' = \mathrm{FHG}(k, X')$. For, if $k = i$, then $e = e'$. On the other hand, if $k \neq i$, then e is dominated by the edit $\mathrm{FHG}\big(i, X \setminus \{e'\}\big)$. (Fellegi and Holt 1976, page 29, item 2, except that Fellegi and Holt required that the edit e be essentially new.)

3. The edit generation step $e = \mathrm{FHG}(i, X)$ need not be calculated if there is a subset X' of X such that the edit $e' = \mathrm{FHG}(i, X')$ is essentially new and has already been calculated. Otherwise the edit e is dominated by the edit e'. (Fellegi and Holt 1976, page 29, item 3.) Hence it is only necessary to use the minimal edit sets for which it is possible to generate an essentially new edit: Chen (1998) gives algorithms to find such minimal subsets.

4. Discard any dominated edits generated at any stage of the edit generation process (Kunnathur 1982, page 39). Otherwise, as spelt out in Lemma 2.4.10, edits generated from the dominated edits will be dominated by edits generated from the dominating edits.

Another method of dealing with the slowness of edit generation is to compromise by not completing the full edit generation process. For example, Barcaroli and Venturi (1993) partition the set of edits to reduce the size of the generated edit set. For arithmetic edits expressed as certain linear inequalities, some approaches are presented by Draper and Winkler (1997) and by Garcia (2003, 2005).

Yet another method of dealing with the slowness of edit generation is to give up altogether on generating all edits prior to the error localisation process, and instead, to treat each erroneous record as a separate stand-alone problem. That is, the method for each erroneous record is to attempt to generate only those edits that will be relevant to finding a useful covering set for that record. Such record-by-record covering set approaches are presented by Garfinkel (1979), Garfinkel, Kunnathur and Liepins (1984, 1986b), Winkler and Chen (2002), and Chen and Winkler (2004). There are similar approaches for arithmetic edits expressed as linear inequalities, presented by Garfinkel, Kunnathur and Liepins (1986a, 1988), Ragsdale and McKeown (1996), de Waal (2003a, Chapter 10), de Waal and Coutinho (2005, Section 9), and Riera-Ledesma and Salazar-González (2007a). The general scheme for such solutions is to first determine whether a smallest covering set of a small set of generated edits (such as of the explicit edits failed by the given record) is an error localisation solution, and if not, to generate additional edits until a smallest covering set is an error localisation solution.

Of course there are also many other record-by-record methods of solving the error localisation problem without using the covering set method or any variations to it. The various solution methods have been described in Chapter 1, Section 1.6.

Most of the record-by-record methods can be seen as a trade-off between search and generation, where search means systematically testing potential solutions and generation means finding new edits or other constraints. Thus the large-scale generation of edits is avoided.

Although the record-by-record methods avoid large-scale edit generation, the workload for each record is increased. In contrast, the idea of the covering set method is to do a large amount of work even before the data are received, with correspondingly less work for each individual record. The covering set method would therefore be best applied to data sets 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 are entered, the error localisation itself should take less time than the record by record approach.

The use of logic also provides some hope. Provided that the edits are suitably formalised in logical terms, logical "consequence finders" can potentially be used to efficiently generate huge numbers of edits. For example, Simon and del Val (2001) present consequence finders that can generate over 10^{70} edits, represented by Binary Decision Diagrams.

In order to use such logical consequence finders, two problems must be solved. Firstly, edit generation functions must be formalised in terms of logic. Such a formalisation is the topic of the next two chapters: Chapter 3 presents some background on logic, while Chapter 4 presents the actual formalisation of edit generation functions. Secondly, it must be demonstrated that the function calculated by the logical consequence finder is in fact covering set correctible, so that the covering set method can be applied. Therefore, in Chapter 5 we present a formalisation of covering set correctibility in terms of some natural constructs of logic. The corresponding results for arithmetic edits are presented in Chapter 7.

While the covering set method seems promising, it has its problems in terms of speed and memory usage. There are many techniques for improving or compromising the covering set method. One method that has not been fully explored is the use of fast consequence finders. In order to use fast consequence finders, all aspects of the covering set method must be formalised in terms of logic. The aspects include both edit generation and covering set correctibility.

2.8 Conclusion

This chapter has introduced the covering set method for data editing, as applied to categorical edits. It has also introduced a range of basic definitions and properties relevant to the covering set method for categorical edits. As for arithmetic edits, we leave the corresponding introduction to Chapter 7.

The main assumptions and concepts introduced in this chapter relate to: data records, edits, error localisation, edit generation functions, the covering set method itself, and properties of edit generation functions. We assume that the data is arranged in records, and that edits are used to specify potential errors in the data and apply to one record at a time. The covering set method solves the error localisation problem:

given a record and a set of edits, find a (smallest) set of fields on which the record can be corrected. The covering set method depends on finding a set of generated edits that are found via an edit generation function, such as the function FH. Provided that the edit generation function is covering set correctible, then the error localisation solutions for a given record $\boldsymbol{v}$ are exactly the covering sets of the generated edits failed by the record $\boldsymbol{v}$.

There are many edit generation functions: those described in this chapter are subfunctions (Definition 1.9.3) of the function FH. Their inter-relationships are presented in Figure 2.2, which shows the function FH and four subfunctions, some of which are subfunctions of each other. The functions MENFH and FCF_ω are subfunctions of the function ENFH, which in turn is a subfunction of FH; and the function MFH is also a subfunction of FH.

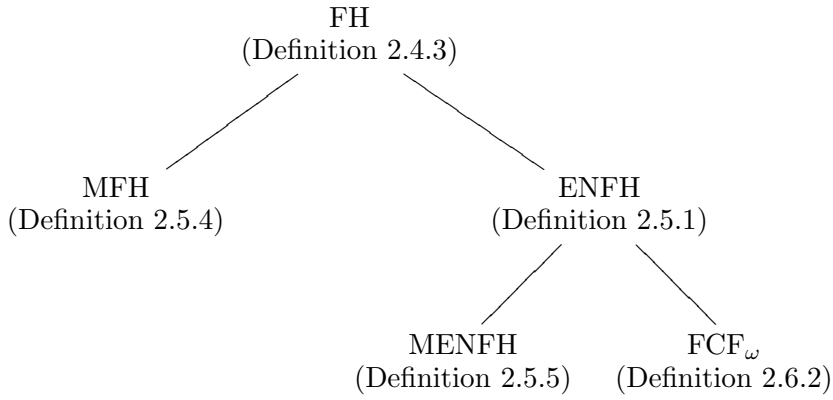

Figure 2.2: Diagram of the inter-relationships amongst the various edit generation functions mentioned in this chapter. The functions are all subfunctions of the function FH. Each function is a subfunction of the function above it in the tree.

Each of the functions described in this chapter, and displayed in Figure 2.2, is covering set correctible and can therefore be used with the covering set method. This chapter contains summaries of the proofs of the covering set correctibility of each of the functions except for the function FCF_ω, which has a complex proof, left until Chapter 6.

The proof of covering set correctibility consists, for all the functions presented here, of two parts, corresponding to the two parts of the definition of covering set correctibility. The two parts are: error correction guarantee and error correction totality.

The proofs of the error correction guarantee of the various functions have mostly been published in the past, although the term "error correction guarantee" had not been defined. The proof of the error correction guarantee for FH and the essence of the proof for ENFH was presented by Fellegi and Holt (1973, 1976). The essence of the proof of the error correction guarantee of the functions MFH and MENFH was presented by Kunnathur (1982). In this chapter we also saw that the error correction guarantee of MFH and MENFH is a consequence of a general result about superior functions (Definition 1.9.4). The proof of the error correction guarantee of FCF_ω is

presented in Chapter 6.

The proofs of error correction totality of the various functions have not been published in the past, partly because the concept of error correction totality had not been defined and partly because the proofs are straightforward. The error correction totality of FH follows quickly from a lemma of Fellegi and Holt (1973, 1976), while the proof of the error correction totality of the other functions follows from a general result about subfunctions.

Since the covering set method is slow and uses much memory, there have been techniques developed for improving or compromising the covering set method. Logic provides some potential for improvement, because of the existence of fast consequence finders. But in order to use such consequence finders it is necessary to formalise in terms of logic the two main components of the covering set method. These two components are: edit generation and covering set correctibility. Most of the remainder of this thesis deals with the formalisation of the two components.

2.9 Appendix: Different statements of the FCF Algorithm

The above algorithm statement (Section 2.6) differs in places from those given in the five papers by Garfinkel, Kunnathur and Liepins (G, K & L 1984, 1986b; Kunnathur 1982; Liepins 1980a, 1984), which differ in places from each other. The differences appear in two aspects: the choice of edits to use in any edit generation, and the domination rules.

Firstly, in the statement given in Section 2.6, the edits used at node $(i_1, \ldots, i_m)$, that is the edits in the set X of Definition 2.6.4, part c, must appear at nodes above $(i_1, \ldots, i_m)$. Liepins (1980a, 1984) appears to agree with this, referring to "antecedent" nodes, each of which is defined however as the single node immediately above the current node. Differently, Kunnathur (1982) and G, K & L (1984, 1986b) require that the edits in the set X are at any nodes already traversed in the forest (Kunnathur) or in the current tree (G, K & L 1984, 1986b). The proofs in Chapter 6 show that Kunnathur's and G, K & L's sets X are bigger than necessary: it is enough to use Liepins' definition of X - but using all antecedent nodes rather than just the immediate antecedent node.

The second difference between the statement of Section 2.6 and those of G, K & L is about the domination rules. The statement of Section 2.6 follows Kunnathur (1982). Differently, G, K & L (1984, 1986b) - albeit ambiguously - and the two papers by Liepins (1980a, 1984) say that newly generated dominated edits should be *discarded* at their nodes, rather than *replaced* by the dominating edits. Instead of the Domination Rule 1 of Definition 2.6.5, they use:

G, K & L Domination Rule 1: Let the current node be σ. Let S_σ be the set of nodes visited prior to visiting the node σ. If $\sigma \neq ()$, then let σ' be the node visited immediately before σ. Calculate $\mathrm{GenV}(\sigma, E, \sigma)$ as follows:

(a) If $\sigma = ()$, then remove all dominated edits from $\mathrm{GenI}(\sigma, E)$. That is,

$$\mathrm{GenV}\big(((), E, ()\big) = \mathrm{Max} \circ \mathrm{GenI}\big(((), E\big).$$

(b) If $\sigma \neq ()$, then, for each α in $\mathrm{GenI}(\sigma, E)$, if there is a node τ in S_σ and an edit β in $\mathrm{GenV}(\tau, E, \sigma')$ with $\beta \supset \alpha$, then exclude the edit α from the set $\mathrm{GenV}(\sigma, E, \sigma)$. That is,

$$\begin{aligned}\mathrm{GenV}(\sigma, E, \sigma) = \big\{\alpha \in \mathrm{GenI}(\sigma, E) \mid & \\ & \text{for each edit } \beta \text{ in } \big(\mathrm{GenI}(\sigma, E) \cup \bigcup_{\tau \in S_\sigma} \mathrm{GenV}(\tau, E, \sigma')\big) \\ & \text{we have that } \alpha \not\subset \beta\big\}.\end{aligned}$$

However, discarding dominated edits can cause a node to become empty, preventing traversal of the subtree with that node as root. Below is an example to show that

discarding a dominated edit, as required by G, K & L Domination Rule 1, would cause the function FCF_ω to not be covering set correctible. In particular FCF_ω would not have the error correction guarantee.

Example 2.9.1. *This is a case where using G, K & L Domination Rule 1 causes the* FCF *function to no longer have the error correction guarantee. Let* $A_1 = A_2 = \{1,2,3\}$. *Let* $A_3 = \{1,2,3,4,5,6,7,8,9\}$. *Let the order of the fields* $= \omega =$ (*field* 1, *field* 2, *field* 3).

Let the set E *of explicit edits be* $\{e_1, e_2, e_3, e_4, e_5, e_6, e_7\}$, *where*

$$\begin{array}{rcccccc} e_1 & = & \{1,2\} & \times & \{2,3\} & \times & \{3,4,9\} \\ e_2 & = & \{3\} & \times & \{2,3\} & \times & \{3,4,8\} \\ e_3 & = & \{1,3\} & \times & \{1\} & \times & \{3,5,6\} \\ e_4 & = & \{2\} & \times & \{1\} & \times & \{3,5,7\} \\ e_5 & = & \{1\} & \times & \{2\} & \times & \{2,3\} \\ e_6 & = & \{1\} & \times & \{1,3\} & \times & \{1,3\} \\ e_7 & = & \{1\} & \times & A_2 & \times & \{1,2,4,5,6,7,8,9\} \end{array} .$$

The calculation of $\mathrm{FCF}_\omega(E)$ *(using Definition 2.6.2 of Section 2.6) is given in Table 2.1. Each cell of the table lists the generated edits at the various stages of the algorithm, and the actual calculation of the generated edits is in the subsequent table, Table 2.2. The first row of Table 2.1 lists all nodes* σ *in order of traversal, except* $\sigma = (1,2,3)$ *and* $\sigma = (1,3)$ *for which* $\mathrm{GenI}(\sigma, E)$ *is empty. The second row lists the values of* $\mathrm{GenI}(\sigma, E)$, *which (other than* $\mathrm{GenI}\big((), E\big)$*) depend on the relevant values of* $\mathrm{GenV}(-, E, -)$. *These latter are given in the remaining rows: each row gives, for some node* τ, *the values of* $\mathrm{GenV}(\sigma, E, \tau)$ *for all the values of* σ *before or equal to* τ. *Each row corresponds to one node* τ, *and the nodes* τ *are in the order of the traversal of the field code forest. Thus the various sets* $\mathrm{GenV}(\sigma, E, \tau)$ *are calculated in order row by row and left to right in each row. The last column gives the edit domination relationships that are used in the table. The last row, corresponding to the last node* (3), *also gives the values of* $\mathrm{GenF}(\sigma, E)$.

From Table 2.1, $$\begin{aligned} \mathrm{FCF}_\omega(E) &= \bigcup \big\{ \mathrm{GenF}(\sigma, E) \mid \sigma \in F(\omega) \big\} \\ &= \{e_1, e_2, e_3, e_4, e_8, e_9, e_{15}, e_{19}\} . \end{aligned}$$

Chapter 6 will show that the covering set method can be successfully used with this set to do error localisation.

If the Domination Rule 1 of Definition 2.6.5 is replaced by G, K & L's Domination Rule 1, we obtain a different set of edits, which we will call $\mathrm{FCF}'_\omega(E)$, *for which the covering set method cannot be successfully applied. The calculation of* $\mathrm{FCF}'_\omega(E)$ *is in Table 2.3, which has the same structure as Table 2.1. The calculations that are changed are circled. The intermediate calculations (rows) at Nodes (), (1), (1,2), (1,2,3) and (1,3) are unchanged. However Node (2) is empty because the edits generated there are dominated by a previous edit* (e_{15}). *This causes Node (2,3) to also be empty.*

Then, when traversing Node (2,3), the edits at Node () are not updated by dominating edits. Then, unlike in Table 2.1, new edits $e_{20}, e_{21}, e_{22}, e_{23}$ *and* e_{24} *can be calculated at Node (3) from the edits at Node () - the calculation is given in Table 2.4.*

From Table 2.3, $\mathrm{FCF}'_\omega(E) = \bigcup\{\mathrm{GenF}(\sigma, E) \mid \sigma \in F(\omega)\}$

$$= \{e_1, e_2, e_3, e_4, e_7, e_8, e_9, e_{15}, e_{20}, e_{23}\}\,.$$

The two sets $\mathrm{FCF}_\omega(E)$ *and* $\mathrm{FCF}'_\omega(E)$ *are the same except that* $\mathrm{FCF}_\omega(E)$ *contains edit* e_{19} *while* $\mathrm{FCF}'_\omega(E)$ *contains edits* e_7, e_{20} *and* e_{23} *. This difference causes a difference in the success of the covering set method.*

We now demonstrate that the function FCF'_ω *does not have the error correction guarantee. We do this by demonstrating that the error in the record* $\boldsymbol{v} = (1, 3, 3)$ *cannot be localised using the covering set method with the edit set* $\mathrm{FCF}'_\omega(E)$*. The edits of* $\mathrm{FCF}'_\omega(E)$ *failed by* $\boldsymbol{v}$ *are* $\{e_1, e_8, e_{15}, e_{20}, e_{23}\}$*. A smallest covering set of this set is the set* $\{$*field* 2, *field* 3$\}$*. However there is no way to correct* $\boldsymbol{v}$ *by changing only fields* 2 *and* 3*, as seen next:*

> *Field* 3 *must take the value* 3*, in order to satisfy edit* e_7 *(since field* 1 *is to remain unchanged).*
>
> *If field* 2 *is changed to the value* 2 *then the new record* $(1, 2, 3)$ *fails edit* e_1 *.*
>
> *If field* 2 *is changed to value* 1 *then the new record* $(1, 1, 3)$ *fails edit* e_3 *.*

However it is possible to use the function FCF_ω *, which does have the error correction guarantee, to correct* $\boldsymbol{v}$*. The edits of* $\mathrm{FCF}_\omega(E)$ *which are failed by* $\boldsymbol{v}$ *are* $\{e_1, e_8, e_{15}, e_{19}\}$ *- the same set as above but with the edit* e_{19} *added and the edits* e_{20} *and* e_{23} *removed. The set* $\{$*field* 2, *field* 3$\}$ *no longer covers this set, but the set* $\{$*field* 1, *field* 3$\}$ *does, and indeed it is possible to correct* $\boldsymbol{v}$ *by changing the values of fields 1 and 3. For example, the record* $(2, 3, 1)$ *satisfies all the edits.*

Table 2.1: *Calculation of* $\mathrm{FCF}_\omega(E)$ *for Example 2.9.1. Each cell of the table lists the generated edits at the various stages of the algorithm, and the actual calculation of the generated edits is in the subsequent table, Table 2.2. The first row of the table below lists all nodes* σ *in order of traversal, except* $\sigma = (1,2,3)$ *and* $\sigma = (1,3)$ *for which* $\mathrm{GenI}(\sigma, E)$ *is empty. The second row lists the values of* $\mathrm{GenI}(\sigma, E)$*, which (other than* $\mathrm{GenI}((), E)$*) depend on the relevant values of* $\mathrm{GenV}(-, E, -)$*. These latter are given in the remaining rows: each row gives, for some node* τ*, the values of* $\mathrm{GenV}(\sigma, E, \tau)$ *for all the values of* σ *before or equal to* τ*. Each row corresponds to one node* τ*, and the nodes* τ *are in the order of the traversal of the field code forest. Thus the various sets* $\mathrm{GenV}(\sigma, E, \tau)$ *are calculated in order row by row and left to right in each row. The last column gives the edit domination relationships that are used in the table. The last row, corresponding to the last node* (3)*, also gives the values of* $\mathrm{GenF}(\sigma, E)$*.*

	$\sigma =$						
	$()$	(1)	$(1,2)$	(2)	$(2,3)$	(3)	*Notes*
$\mathrm{GenI}(\sigma, E)$	$\{e_1, e_2, e_3, e_4, e_5, e_6, e_7\}$	$\{e_8, e_9, e_{10}, e_{11}, e_{12}, e_{13}, e_{14}\}$	$\{e_{15}\}$	$\{e_{16}, e_{17}, e_{18}\}$	$\{e_{19}\}$	$\emptyset$	
$\mathrm{GenV}(\sigma, E, ())$	$\{e_1, e_2, e_3, e_4, e_5, e_6, e_7\}$						
$\mathrm{GenV}(\sigma, E, (1))$	$\{e_1, e_2, e_3, e_4, e_5, e_6, e_7\}$	$\{e_8, e_9\}$					$e_8 \supseteq e_{10}$ $e_8 \supseteq e_{11}$ $e_8 \supseteq e_{12}$ $e_9 \supseteq e_{13}$ $e_9 \supseteq e_{14}$
$\mathrm{GenV}(\sigma, E, (1,2))$	$\{e_1, e_2, e_3, e_4, e_5, e_6, e_7\}$	$\{e_8, e_9\}$	$\{e_{15}\}$				
$\mathrm{GenV}(\sigma, E, (2))$	$\{e_1, e_2, e_3, e_4, e_5, e_6, e_7\}$	$\{e_8, e_9\}$	$\{e_{15}\}$	$\{e_{15}\}$			$e_{15} \supseteq e_{16}$ $e_{15} \supseteq e_{17}$ $e_{15} \supseteq e_{18}$
$\mathrm{GenV}(\sigma, E, (2,3))$	$\{e_1, e_2, e_3, e_4, e_{19}\}$	$\{e_8, e_9\}$	$\{e_{15}\}$	$\{e_{15}\}$	$\{e_{19}\}$		$e_{19} \supseteq e_5$ $e_{19} \supseteq e_6$ $e_{19} \supseteq e_7$
$\mathrm{GenV}(\sigma, E, (3))$ $= \mathrm{GenF}(\sigma, E)$	$\{e_1, e_2, e_3, e_4, e_{19}\}$	$\{e_8, e_9\}$	$\{e_{15}\}$	$\{e_{15}\}$	$\{e_{19}\}$	$\emptyset$	

Table 2.2: *Calculation of generated edits used in Table 2.1 for the calculation of* $\mathrm{FCF}_\omega(E)$ *for Example 2.9.1.*

$e_8 =$	$\mathrm{FHG}\big(1, \{e_1, e_2\}\big)$	$=$	A_1	$\times$	$\{2,3\}$	$\times$	$\{3,4\}$
$e_9 =$	$\mathrm{FHG}\big(1, \{e_3, e_4\}\big)$	$=$	A_1	$\times$	$\{1\}$	$\times$	$\{3,5\}$
$e_{10} =$	$\mathrm{FHG}\big(1, \{e_1, e_2, e_5\}\big)$	$=$	A_1	$\times$	$\{2\}$	$\times$	$\{3\}$
$e_{11} =$	$\mathrm{FHG}\big(1, \{e_1, e_2, e_6\}\big)$	$=$	A_1	$\times$	$\{3\}$	$\times$	$\{3\}$
$e_{12} =$	$\mathrm{FHG}\big(1, \{e_1, e_2, e_7\}\big)$	$=$	A_1	$\times$	$\{2\}$	$\times$	$\{4\}$
$e_{13} =$	$\mathrm{FHG}\big(1, \{e_3, e_4, e_6\}\big)$	$=$	A_1	$\times$	$\{1\}$	$\times$	$\{3\}$
$e_{14} =$	$\mathrm{FHG}\big(1, \{e_3, e_4, e_7\}\big)$	$=$	A_1	$\times$	$\{1\}$	$\times$	$\{5\}$
$e_{15} =$	$\mathrm{FHG}\big(2, \{e_8, e_9\}\big)$	$=$	A_1	$\times$	A_2	$\times$	$\{3\}$
$e_{16} =$	$\mathrm{FHG}\big(2, \{e_1, e_3\}\big)$	$=$	$\{1\}$	$\times$	A_2	$\times$	$\{3\}$
(Also, $e_{16} = \mathrm{FHG}(2, \{e_1, e_6\}) = \mathrm{FHG}(2, \{e_5, e_6\}) = \mathrm{FHG}(2, \{e_1, e_5, e_6\})$ $= \mathrm{FHG}(2, \{e_3, e_5, e_6\}) = \mathrm{FHG}(2, \{e_1, e_3, e_5\})$ $= \mathrm{FHG}(2, \{e_1, e_3, e_6\}) = \mathrm{FHG}(2, \{e_1, e_3, e_5, e_6\})$ *.)*							
$e_{17} =$	$\mathrm{FHG}\big(2, \{e_1, e_4\}\big)$	$=$	$\{2\}$	$\times$	A_2	$\times$	$\{3\}$
$e_{18} =$	$\mathrm{FHG}\big(2, \{e_2, e_3\}\big)$	$=$	$\{3\}$	$\times$	A_2	$\times$	$\{3\}$
$e_{19} =$	$\mathrm{FHG}\big(3, \{e_7, e_{15}\}\big)$	$=$	$\{1\}$	$\times$	A_2	$\times$	A_3

Table 2.3: *Calculation, using G, K & L Domination Rule 1, of* $\mathrm{FCF}'_\omega(E)$ *for Example 2.9.1. This table has been calculated in order to compare* $\mathrm{FCF}'_\omega(E)$ *with* $\mathrm{FCF}_\omega(E)$*, which was given in Table 2.1. The table has the same structure as Table 2.1. The entries that are different from Table 2.1 are circled.*

	$\sigma =$ $()$	(1)	$(1,2)$	(2)	$(2,3)$	(3)	*Notes*
$\mathrm{GenI}(\sigma, E)$	$\{e_1, e_2, e_3, e_4, e_5, e_6, e_7\}$	$\{e_8, e_9, e_{10}, e_{11}, e_{12}, e_{13}, e_{14}\}$	$\{e_{15}\}$	$\{e_{16}, e_{17}, e_{18}\}$	$\boxed{\emptyset}$	$\{e_{20}, e_{21}, e_{22}, e_{23}, e_{24}\}$	
$\mathrm{GenV}\big(\sigma, E, ()\big)$	$\{e_1, e_2, e_3, e_4, e_5, e_6, e_7\}$						
$\mathrm{GenV}\big(\sigma, E, (1)\big)$	$\{e_1, e_2, e_3, e_4, e_5, e_6, e_7\}$	$\{e_8, e_9\}$					$e_8 \supseteq e_{10}$ $e_8 \supseteq e_{11}$ $e_8 \supseteq e_{12}$ $e_9 \supseteq e_{13}$ $e_9 \supseteq e_{14}$
$\mathrm{GenV}\big(\sigma, E, (1,2)\big)$	$\{e_1, e_2, e_3, e_4, e_5, e_6, e_7\}$	$\{e_8, e_9\}$	$\{e_{15}\}$				
$\mathrm{GenV}\big(\sigma, E, (2)\big)$	$\{e_1, e_2, e_3, e_4, e_5, e_6, e_7\}$	$\{e_8, e_9\}$	$\{e_{15}\}$	$\boxed{\emptyset}$			$e_{15} \supseteq e_{16}$ $e_{15} \supseteq e_{17}$ $e_{15} \supseteq e_{18}$
$\mathrm{GenV}\big(\sigma, E, (2,3)\big)$	$\{e_1, e_2, e_3, e_4, e_5, e_6, e_7\}$	$\{e_8, e_9\}$	$\{e_{15}\}$	$\boxed{\emptyset}$	$\boxed{\emptyset}$		*no domination relations*
$\mathrm{GenV}\big(\sigma, E, (3)\big)$	$\{e_1, e_2, e_3, e_4, e_{20}, e_{23}, e_7\}$	$\{e_8, e_9\}$	$\{e_{15}\}$	$\boxed{\emptyset}$	$\boxed{\emptyset}$	$\{e_{20}, e_{23}\}$	$e_{20} \supseteq e_5$ $e_{23} \supseteq e_6$ $e_{23} \supseteq e_{21}$ $e_{20} \supseteq e_{22}$ $e_{20} \supseteq e_{24}$ $e_{23} \supseteq e_{24}$
$= \mathrm{GenF}(\sigma, E)$							

Table 2.4: *Calculation of additional generated edits used in Table 2.3 for the calculation of* $\mathrm{FCF}'_\omega(E)$ *for Example 2.9.1. The calculations for the other generated edits are in Table 2.2.*

$e_{20} =$	$\mathrm{FHG}\big(3, \{e_1, e_7\}\big)$	$=$	$\{1\}$	$\times$	$\{2,3\}$	$\times$	A_3
$e_{21} =$	$\mathrm{FHG}\big(3, \{e_3, e_7\}\big)$	$=$	$\{1\}$	$\times$	$\{1\}$	$\times$	A_3
(Also, $e_{21} = \mathrm{FHG}(3, \{e_3, e_6, e_7\})$ *.)*							
$e_{22} =$	$\mathrm{FHG}\big(3, \{e_5, e_7\}\big)$	$=$	$\{1\}$	$\times$	$\{2\}$	$\times$	A_3
(Also, $e_{22} = \mathrm{FHG}(3, \{e_1, e_5, e_7\})$ *.)*							
$e_{23} =$	$\mathrm{FHG}\big(3, \{e_6, e_7\}\big)$	$=$	$\{1\}$	$\times$	$\{1,3\}$	$\times$	A_3
$e_{24} =$	$\mathrm{FHG}\big(3, \{e_1, e_6, e_7\}\big)$	$=$	$\{1\}$	$\times$	$\{3\}$	$\times$	A_3