A Simple Branching Scheme for Solving the Error Localisation Problem

Discussion paper 03010

Ton de Waal

Statistics Netherlands

Voorburg/Heerlen, November 2003

Explanation of symbols

.	= data not available
*	= provisional figure
x	= publication prohibited (confidential figure)
–	= nil or less than half of unit concerned
–	= (between two figures) inclusive
0 (0,0)	= less than half of unit concerned
blank	= not applicable
2002–2003	= 2002 to 2003 inclusive
2002/2003	= average of 2002 up to and including 2003
2002/’03	= crop year, financial year, school year etc. beginning in 2002 and ending in 2003

Due to rounding, some totals may not correspond with the sum of the separate figures.

Publisher
Statistics Netherlands
Prinses Beatrixlaan 428
2273 XZ Voorburg
The Netherlands

Printed by
Statistics Netherlands - Facility Services

Cover design
WAT ontwerpers, Utrecht

Information
E-mail: infoservice@cbs.nl

Where to order
E-mail: verkoop@cbs.nl

Internet
http://www.cbs.nl

ISSN: 1572-0314
Key figure: X-10
Production code: 6008303010

Statistics Netherlands

A SIMPLE BRANCHING SCHEME FOR SOLVING THE ERROR LOCALISATION PROBLEM

Summary: In this paper we present a new algorithm for solving the error localisation problem for a mix of continuous and categorical data. This algorithm is based on constructing a binary search tree. In this tree continuous variables are treated by fixing them to their original value or by eliminating them by means of an extension of Fourier-Motkzin elimination. Categorical variables are treated by fixing them to the most-likely value or by excluding (the equivalence class of) this value from the current domain. We also present an extension to include integer-valued data, and a heuristic for quickly determining (suboptimal) solutions to the error localisation problem.

Keywords: branch-and-bound, error localisation, Fellegi-Holt paradigm, Fourier-Motkzin elimination, statistical data editing

1. Introduction

The so-called error localisation problem in automatic editing has received ample attention at Statistics Netherlands. In particular, we have focussed our attention on developing new algorithms based on the Fellegi-Holt paradigm of minimum change. This paradigm says that as few fields as possible should be changed so that all edit checks (or edits for short) become satisfied.

In recent years we have developed and implemented a branch-and-bound algorithm for solving the error localisation problem in a mix of categorical and continuous data (see, De Waal and Quere, 2003). This algorithm has later been extended to include integer-valued data (see De Waal, 2003). The basic idea of the developed algorithm for a mix of categorical and continuous data is that a binary tree is constructed (see Figure 1 in Section 4 for an example of a binary tree). In each node of this tree a variable is selected that has not yet been selected in any predecessor node. All continuous variables are selected before any categorical variable is. If all variables have been selected in a predecessor node, we have reached a terminal node of the tree. After selection of a variable two branches are constructed: in one branch the selected variable is fixed to its original value, in the other branch the selected variable is eliminated from the set of current edits. Fixing a variable to its original value corresponds to assuming that this original value is correct, eliminating a variable from the set of current edits corresponds to assuming that the original value of this variable is incorrect and has to be modified. Eliminating a variable is a relatively complicated process. It amounts to generating a set of implied edits that do not involve this variable. That set of implied edits becomes the set of edits corresponding to the current branch of the tree. If a continuous variable is to be eliminated, we basically apply an extension of Fourier-Motzkin elimination (see

Duffin, 1974; De Waal and Quere, 2003) to eliminate that variable from the set of edits. To eliminate a categorical variable from the set of edits, we basically apply the method of Fellegi and Holt to generate implied edits (see Fellegi and Holt, 1976; De Waal and Quere, 2003). If and only if the edits corresponding to a certain node are satisfied by the values of the remaining variables, the variables that have been eliminated in order to arrive at that node form a (possibly suboptimal) solution to the error localisation problem.

The Fellegi-Holt method to eliminate a categorical variable is a generalisation of the (ground) resolution procedure known from propositional logic (see, e.g., Chandru and Hooker, 1999; Hooker, 2000). The resolution procedure is a classical method to test whether a number of so-called clauses can be satisfied by giving appropriate truth values to Boolean variables. The Fellegi-Holt procedure is a method to test whether a number of categorical edits can be satisfied by giving appropriate values to the categorical variables. In the mathematical logic community, the Fellegi-Holt procedure would be considered as (a form of) multivalent resolution.

Another classical procedure in propositional logic to test whether a set of clauses can be satisfied is the so-called Putnam-Davis (or: Putnam-Davis-Loveland) procedure. In this paper we modify the previously proposed algorithm for solving the error localisation problem by replacing the Fellegi-Holt method for eliminating categorical variables with a procedure similar to the Davis-Putnam procedure. Only the treatment of categorical data distinguishes the new algorithm from the previously proposed algorithm. For purely continuous data, the two algorithms are exactly the same.

The algorithm we propose in the present paper finds all optimal solutions to an instance of the error localisation problem in a mix of categorical and continuous data, given a user-specified maximum for the number of values $N_{\max}$ that have to be changed. If there several optimal solutions, later one of them can be selected by means of a secondary, more statistical criterion.

In case more than $N_{\max}$ values in a record need to be changed, the record is not handled automatically by means of this algorithm. There are two reasons for introducing the upper bound $N_{\max}$. The first one is a methodological reason: if many values in a record need to be changed, we feel that one should be extremely careful with the application of automatic error localisation. In principle, we feel that such records should be edited interactively instead of automatically, because the quality of these records when edited automatically is likely to be low. Note, however, that for non-influential records one might make an exception, because the impact of these records on the final publication figures is low anyway. Automatic edit and imputation of such non-influential records will at least make them internally consistent, which helps during further processing of the data. Another reason for introducing the upper bound $N_{\max}$ is a practical one: the computing time of the exact algorithm we propose in this paper becomes too high for practical application on records containing many errors. As we argued above, we feel that automatic edit and imputation should either be applied to records for which at most $N_{\max}$ values need to be changed, or to non-influential records. For such non-influential records it is, however, not vital to apply an exact algorithm. A heuristic that yields a good, not necessarily optimal, solution seems sufficient for such records. The developed algorithm can be used as a basis for such a heuristic.

A mathematical formulation for the error localisation problem based on the Fellegi-Holt paradigm of minimum change is given in Section 2. Section 3 discusses the elimination technique for continuous data on which the proposed algorithm is based, and Section 4 the developed algorithm itself. An example illustrating the algorithm is given in Section 5. Section 6 extends the algorithm to integer-valued data. Section 7 discusses our heuristic algorithm. Finally, Section 8 concludes the paper with a brief discussion.

2. Mathematical formulation of the error localisation problem

We denote the values of the categorical variables by v_i ($i = 1,\dots,m$) and the values of the numerical variables by x_k ($k = 1,\dots,n$). The index set of the integer-valued variables is denoted by I. The remaining numerical variables are continuous, which in this paper means that they are rational. For categorical data we denote the domain, i.e. the set of the possible values, of variable i by D_i. We assume that every edit j ($j = 1,\dots,J$) is written in either of the two following forms:

$$\text{IF } v_i \in F_i^j \ (i = 1,\dots,m) \text{ THEN } (x_1,\dots,x_n) \in \{\mathbf{x} \mid a_{1j}x_1 + \dots + a_{nj}x_n + b_j \geq 0\}, \tag{1a}$$

or

$$\text{IF } v_i \in F_i^j \ (i = 1,\dots,m) \text{ THEN } (x_1,\dots,x_n) \in \{\mathbf{x} \mid a_{1j}x_1 + \dots + a_{nj}x_n + b_j = 0\}, \tag{1b}$$

where $F_i^j \subseteq D_i$. That is, edit j ($j = 1,\dots,J$) is satisfied by a record $(v_1,\dots,v_m,x_1,\dots,x_n)$ if (1a), respectively (1b) holds. In (1) the a_{kj} are assumed to be rational numbers.

The condition after the IF-statement, i.e. "$v_i \in F_i^j$ for all $i = 1,\dots,m$", is called the IF-condition of the edit. The numerical condition after the THEN-statement is called the THEN-condition. If the IF-condition does not hold true for a particular record, the edit is always satisfied, irrespective of the values of the numerical variables. A categorical variable i is said to *enter*, or to be *involved in*, an edit j given by (1) if $F_i^j \subset D_i$ and $F_i^j \neq D_i$, i.e. if F_i^j is strictly contained in the domain of variable i. That edit is then said to be *involved with* this categorical variable. A numerical variable k is said to *enter*, or to be *involved in*, the THEN-condition of edit j given by (1) if $a_{kj} \neq 0$. That THEN-condition is then said to be *involved with* this numerical variable.

By multiplying b_j and all coefficients a_{kj} ($k = 1,\dots,n$; $j = 1,\dots,J$) involved in a THEN-condition by an appropriately chosen number we can ensure that in each

THEN-condition these coefficients become integral and that their greatest common divisor equals 1.

All edits given by (1) and all integrality constraints, defined by index set *I*, have to be satisfied simultaneously. We assume that the edits and integrality constraints can indeed be satisfied simultaneously.

In many practical cases, certain kinds of missing values are acceptable, for instance when the corresponding questions are not applicable to a particular respondent. We assume that for categorical variables such acceptable missing values are coded by special values, for instance the value "NA" (non-applicable), in their domains. These special values are treated just like an ordinary value. Non-acceptable missing values of categorical variables are not coded. The optimisation problem formulated below will identify these latter missing values as being erroneous. We also assume that numerical THEN-conditions are only triggered if the value of none of the variables involved may be missing. Hence, if for a certain record a THEN-condition involving a variable of which the value is missing is triggered by the categorical values, then either the missing value is erroneous or at least one of the categorical values.

If the set in the THEN-condition of (1) is the entire *n*-dimensional real vector space, then the edit is always satisfied. Such an edit may be discarded. If the set in the THEN-condition of (1) is empty, then the edit is failed by any record for which the IF-condition holds, i.e. for any record for which $v_i \in F_i^j$ for all $i = 1,\ldots,m$. Such an edit may be considered as a purely categorical edit. An edit for which $F_i^j = \emptyset$ for some *i* is by definition always satisfied. An edit for which $F_i^j = D_i$ for all $i = 1,\ldots,m$ is satisfied if and only if the numerical THEN-condition is satisfied. Such an edit may be considered as a purely numerical one.

For each record $(v_1^0,...,v_m^0,x_1^0,...,x_n^0)$ in the data set that is to be edited automatically we now wish to determine, or more precisely: wish to ensure the existence of, a synthetic record $(v_1,...,v_m,x_1,...,x_n)$ such that (1) becomes satisfied for all edits $j = 1,\ldots,J$, the integrality constraints defined by index set *I* become satisfied, and

$$\sum_{i=1}^{m} w_i^c \delta(v_i^0, v_i) + \sum_{k=1}^{n} w_k^r \delta(x_k^0, x_k) \qquad (2)$$

is minimised. Here w_i^c is the non-negative reliability weight of categorical variable *i* ($i = 1,\ldots,m$), w_k^r the non-negative reliability weight of numerical variable *k* ($k = 1,\ldots,n$), $\delta(y^0, y) = 1$ if $y^0 \neq y$ or if y^0 is missing, and $\delta(y^0, y) = 0$ if $y^0 = y$. The reliability weight of a variable expresses how reliable one considers the values of this variable to be. The variables of which the values in the synthetic record differ from the original values together with the variables of which the values were missing form an optimal solution to the error localisation problem. We aim to find all optimal solutions to the error localisation problem.

3. Elimination of a continuous variable

If a continuous variable is to be eliminated, we basically apply Fourier-Motzkin elimination (see Duffin, 1974; De Waal and Quere, 2003) to eliminate that variable from the set of edits. Some care has to be taken in order to ensure that the IF-conditions of the resulting edits are correctly defined. In particular, if we want to eliminate a continuous variable x_r from the current set of edits, we start by copying all edits not involving this continuous variable from the current set of edits to the new set of edits. Next, we consider all edits in format (1) involving x_r pair-wise. Suppose we consider the pair consisting of edit s and edit t. We start by checking whether the intersection of the IF-conditions is non-empty, i.e. whether the intersections $F_i^s \cap F_i^t$ are non-empty for all $i = 1,\dots,m$. If any of these intersections is empty, we do not have to consider this particular combination of edits anymore. So, suppose that all intersections are non-empty.

If the THEN-condition of edit s is an equality, we use the equality

$$x_r = -\frac{1}{a_{rs}}\left(b_s + \sum_{k\neq r} a_{ks}x_k\right)$$

to eliminate x_r from the THEN-condition of edit t. Similarly, if the THEN-condition of edit s is an inequality and the THEN-condition of edit t is an equality, the equality in the THEN-condition of edit t is used to eliminate x_r.

If the THEN-conditions of both edit s and edit t are inequalities, we check whether the coefficients of x_r in those inequalities have opposite signs, i.e. we check whether $a_{rs} \times a_{rt} < 0$. If that is not the case, we do not consider this particular combination of edits anymore. If the coefficients of x_r do have opposite signs, we generate the THEN-condition:

$$(x_1,\dots,x_n) \in \{\mathrm{x} \mid \tilde{a}_1 x_1 + \dots + \tilde{a}_n x_n + \tilde{b} \geq 0\}\,, \tag{3}$$

where

$$\tilde{a}_i = |a_{rs}| \times a_{kt} + |a_{rt}| \times a_{ks} \text{ for all } k = 1,\dots,n \tag{4}$$

and

$$\tilde{b} = |a_{rs}| \times b_t + |a_{rt}| \times b_s\,.$$

Note that x_r indeed does not enter the resulting THEN-condition. This is the THEN-condition of a new implied edit. The IF-condition of this implied edit is given by the intersections $F_i^s \cap F_i^t$ for all $i = 1,\dots,m$.

Equalities in THEN-conditions can be handled more efficiently than as described so far. For instance, if the continuous variable to be eliminated is involved in an equality that has to hold true irrespective of the values of the categorical variables, i.e. is involved in an edit of the following type

$$\text{IF } v_i \in D_i \text{ (for } i = 1,\ldots,m) \text{ THEN } (x_1,\ldots,x_n) \in \{\mathbf{x} \mid a_{1j}x_1 + \ldots + a_{nj}x_n + b_j = 0\}, \qquad (5)$$

then we do not have to consider all edits pair-wise in order to eliminate this variable. Instead, we only have to combine (5) with all other current edits. So, if there are J current edits, we do not have to consider $J(J-1)$ pairs, but only $J-1$ pairs. Besides, the number of resulting implied edits is generally less than when all pairs of current edits are considered. We refer to this rule as the equality-elimination rule.

4. A branching scheme for a mix of categorical and continuous data

The algorithm proposed by De Waal and Quere (2003) succeeds in finding all optimal solutions to the error localisation problem for a mix of categorical and continuous data, i.e. all possible ways of changing a minimum weighted number of variables such that all edits can be satisfied. Unfortunately, the proposed algorithm has two drawbacks. First, when a categorical variable is eliminated, the generation of implied edits may be slow because the number of implicit edits may be very high and, in particular, because the generation process requires many computations. Second, categorical variables may only be treated once all continuous ones have been treated. Due to this latter drawback, the algorithm becomes slow if optimal solutions to the error localisation problem involve categorical variables since much time is spent on examining continuous variables that are later considered correct. To overcome these drawbacks we propose a new algorithm where categorical variables are not handled by means of elimination, and that allows treating categorical variables before all continuous ones have been treated.

4.1 Outline of the algorithm

The basic idea of the new algorithm is that all possible combinations of variables are examined in a smartly constructed sequence. For continuous variables, we basically use the algorithm of De Waal and Quere (2003) to find all optimal solutions to the error localisation problem. We now extend that algorithm to include categorical data. This extension is achieved by searching the domains of the categorical variables. Our search, which is exhaustive, is carried out by constructing a binary search tree. Since the construction of such a binary tree is very time-consuming, we include several shortcuts for pruning the tree, thereby shortening our search. For instance, we do not explore branches of the search tree that are inconsistent or that can only result in less optimal solutions than those already obtained in previously examined branches. We also do not explore branches that require more than $N_{\max}$ variables to be changed. Furthermore, we 'trim' the domains of the categorical variables. This is done when we find a certain value of a categorical variable to be impossible, and also if we do not allow a categorical variable to attain a certain value in subsequent branches of the tree. All values of the categorical variable under consideration that occur in exactly the same subset of the current edits as this value, say v_0, are then excluded from its domain. Such a set of values is referred to as an *equivalence class of* v_0 (*with respect to the set of current edits*). Equivalence classes occur often in realistic edit sets. They allow one to handle several values of a categorical variable simultaneously. We now elaborate upon the basic idea we have sketched in this paragraph.

In each node of the binary tree a variable is selected that has not yet been selected in any predecessor node and that is the most likely one to be incorrect. If all variables have already been selected in a predecessor node, we have reached a terminal node of the tree. During the execution of the algorithm the current domains of the categorical variables are updated. The current domain of a categorical variable in the root node is its original domain. At the start of the algorithm we set the current objective value to zero. After the selection of a variable two branches are constructed. If a continuous variable is selected, the variable is first eliminated by means of the elimination technique described in Section 3 in one branch and later fixed to its original value in the other branch. Eliminating a continuous variable corresponds to assuming that the original value is incorrect, fixing a continuous variable to its original value corresponds to assuming that the original value is correct. After a continuous variable has been eliminated, the objective value of the corresponding branch is updated by adding the reliability weight of the variable under consideration to the current objective value. A continuous variable of which the value is missing can obviously only be eliminated and not fixed to its original value. If a categorical variable is selected, the variable is first fixed to a potentially correct value v_0 in one branch, and the equivalence class of v_0 is later excluded from the current domain of this variable in the other branch. Fixing a categorical variable to a potentially correct value v_0 corresponds to assuming that v_0 is the correct value. Excluding the equivalence class of v_0 from the current domain of a categorical variable, corresponds to assuming that none of the values in the equivalence class of v_0 is correct.

Figure 1 shows a small binary tree involving two categorical variables V_1 and V_2, and one continuous variable X_1. In the root node, N_1, the categorical variable V_1 is selected. In one branch V_1 is fixed to its original value, in the other the domain of V_1 is updated. The child nodes of N_1 are N_2 and N_9. In node N_2, the continuous variable X_1 is selected. In one branch X_1 is fixed to its original value, in the other X_1 is eliminated from the set of edits in node N_2. The terminal nodes of the tree are nodes N_4, N_5, N_7, N_8, N_{11}, N_{12}, N_{14}, and N_{15}.

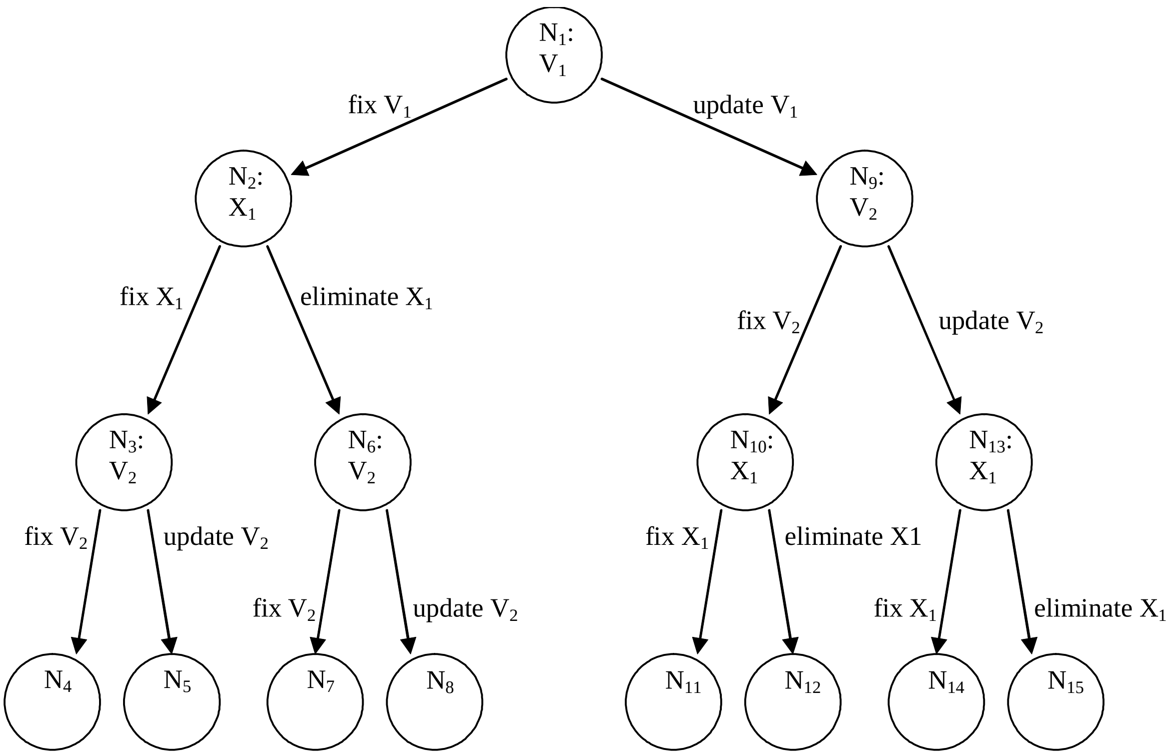

Figure 1. A binary tree

To fix a continuous or categorical variable to a value we simply substitute this value in all current edits. In the new node the fixed variable is removed, and we are left with a problem involving fewer variables. As a result of fixing the selected variable to a value some edits may become satisfied, for instance when a categorical variable is fixed to a value such that the IF-condition of an edit can never become true anymore. These edits may be discarded from the new set of edits. Conversely, some edits not involving any unknowns may be generated that are violated, for instance "$0 \geq 1$". Such an edit can, for instance, arise if a variable x_1 has to satisfy the edit $x_1 \geq 1$, the original value of x_1 equals 0, and we fix x_1 to its original value. If such violated edits not involving any unknowns arise, this branch of the binary tree cannot result in a solution to the error localisation problem.

Since we want to change a minimum weighted number of variables, we determine the value to which a categorical variable is fixed by estimating the sum of the reliability weights of the variables that still need to be changed for each value in its current domain. We select a value v_0 in the current domain of the selected categorical variable for which this estimate is minimal, and fix the categorical variable to this value. By selecting a value for which the estimated sum of the reliability weights of the variables that still need to be changed is minimal, we aim to construct a good solution to the error localisation problem as soon as possible. If the equivalence class of v_0 does not contain the original value of the categorical variable under consideration and the objective value has not yet been increased as a result of updating the domain of this variable, we add the reliability weight of this variable to the current objective value. If a categorical variable is fixed to a value v_0

we delete all edits not involving this value as these edits cannot be triggered in this branch of the tree.

If a value v_0 is excluded from the current domain $D_{\text{cur},i}$ of a categorical variable i, we define the new current domain by $D'_{\text{cur},i} = D_{\text{cur},i} - D(v_0)$, where $D(v_0)$ is the equivalence class of v_0 with respect to the set of current edits. We copy the current edits that involve at least one category of the selected variable that does not occur in $D(v_0)$, and remove all categories of the selected variable that occur in $D(v_0)$ from the current edits. In this manner we obtain a new set of current edits not involving any category in $D(v_0)$. The domains of the categorical variables other than variable i are not updated. If $D(v_0)$ does not include the original value of the selected categorical variable and the objective value has not yet been increased as a result of updating the domain of this variable, the objective value of the corresponding branch is updated by adding the reliability weight of the variable under consideration to the current objective value.

After a variable has been treated in any of the ways above, we can apply constraint propagation techniques. Such techniques could be especially beneficial if a categorical variable has been handled, either by fixing it to a specific value or by excluding some values from its domain. They may enable one to conclude sooner that a certain branch of the search tree cannot yield a feasible solution, or may enable one to find a good feasible solution sooner. Both the ability to conclude quickly that a branch cannot lead to a feasible solution and the ability to find a good solution quickly allow one to prune the search tree, and hence speed up the search for optimal solutions. For more information on constraint propagation techniques we refer to Chandru and Hooker (1999), and Hooker (2000).

If the domain of a variable forms a single equivalence class with respect to the current set of edits, we delete this variable from the set of edits. If the original value of the variable under consideration is not an element of this domain and the objective value has not yet been increased as a result of updating the domain of this variable, we update the objective value by adding the reliability weight of the variable under consideration to the current objective value. If the current domain of any categorical variable becomes empty, we can prune the corresponding branch from the tree because this branch cannot lead to a solution of the error localisation problem.

After we have either eliminated a continuous variable or fixed a categorical one to its most likely value, we check whether the resulting set of edits is satisfied by the original values of the remaining variables. If this is the case, we have found, a possibly suboptimal, solution to the error localisation problem. This is the content of Theorem 1 below. The current objective value equals the value of the objective function (2) for that solution. Such a solution is possibly suboptimal rather than optimal, because earlier on in the binary tree some suboptimal choices may have been made. Whether or not a possibly suboptimal solution is optimal or not, becomes clear once a larger part of the binary tree has been explored.

An interesting aspect of our search tree is that each node defines an associated error localisation problem, namely the problem of changing a minimum (weighted) number of the remaining variables, given the choices that have been made earlier in the search tree. The error localisation problem associated to an internal node is solved in the same manner as the error localisation problem associated to the root node, i.e. the actual error localisation problem for the record under consideration.

In order to prove Theorem 1, which is stated further on in this section, we first state and prove a lemma. The proof of this lemma is essentially a slightly simplified version of the proof of Theorem 1 in De Waal and Quere (2003), or equivalently the proof of Theorem 8.1 in De Waal (2003). We repeat it here in order to make the present paper self-contained.

Lemma. A set of edits corresponding to a certain node can only be satisfied if the set of edits corresponding to its parent node can be satisfied.

Proof. Denote the set of edits and the variables of the node under consideration by Ω_1, respectively T_1. Furthermore, denote the set of edits and the variables of the parent node by Ω_0, respectively T_0.

We have to distinguish between several cases. First, let us suppose that the selected variable is fixed to a value. This is a trivial case. It is clear that if there exist values u_i for $i \in T_1$ that satisfy the edits in Ω_1, there exist values u_i for $i \in T_0$ that satisfy the edits in Ω_0. Namely, for the fixed variable x_r we set the value u_r equal to the value to which variable x_r was fixed.

Let us now suppose that the domain of a categorical variable is reduced by excluding the equivalence class of a selected value. If there exist values u_i for $i \in T_1$ that satisfy the edits in Ω_1, the same values u_i for $i \in T_0$ satisfy the edits in Ω_0.

Finally, let us suppose that a continuous variable x_r has been eliminated. Suppose that there exist values u_i for $i \in T_1$ that satisfy the edits in Ω_1. Each edit in Ω_1 is either obtained from copying the edits in Ω_0 not involving variable x_r, or from two edits in Ω_0 involving variable x_r that have been combined.

It is clear that if the edits in Ω_1 that have been obtained from copying the edits in Ω_0 not involving variable x_r are satisfied by the values u_i for $i \in T_1$, these edits in Ω_0 are also satisfied by the same values for $i \in T_0$.

It remains to prove that if the edits in Ω_1 that have been obtained by combining two edits in Ω_0 are satisfied by $i \in T_1$, there exists a value for variable x_r such that all edits in Ω_0 involving variable x_r can be satisfied. We start by substituting the values u_i for $i \in T_1$ into the edits in Ω_0. As a result, we obtain a number of constraints for the value of the selected variable x_r. Such a constraint can be an equality involving

x_r, a lower bound on x_r, or an upper bound on x_r. That is, these constraints are given by:

$$x_r = M_j^E \tag{6}$$

$$x_r \geq M_j^L\ , \tag{7}$$

and

$$x_r \leq M_j^U\ , \tag{8}$$

where M_j^E , M_j^L and M_j^U are certain constants.

A constraint of type (6) has been obtained from an edit j in Ω_0 of which the THEN-condition can be written in the following form

$$x_r = \sum_{i \neq r} a'_{ij} x_i + b'_j \tag{9}$$

by filling in the values u_i for $i \in T_1$. Similarly, constraints of types (7) and (8) have been obtained from edits in Ω_0 of which the THEN-conditions can be written in the following forms

$$x_r \geq \sum_{i \neq r} a'_{ij} x_i + b'_j$$

and

$$x_r \leq \sum_{i \neq r} a'_{ij} x_i + b'_j\ ,$$

respectively, by filling in the values u_i for $i \in T_1$.

If the constraints given by (6) to (8) do not contradict each other, we can find a value for variable x_r such that this value plus the values u_i for $i \in T_1$ satisfy the edits in Ω_0.

So, suppose the constraints given by (6) to (8) contradict each other. These constraints can only contradict each other if there are constraints s and t given by

1. $x_r = M_s^E$ and $x_r = M_t^E$ with $M_s^E \neq M_t^E$,
2. $x_r = M_s^E$ and $x_r \geq M_t^L$ with $M_s^E < M_t^L$,
3. $x_r \leq M_s^U$ and $x_r = M_t^E$ with $M_s^U < M_t^E$, or
4. $x_r \leq M_s^U$ and $x_r \geq M_t^L$ with $M_s^U < M_t^L$

In case 1 constraints s and t have been derived from edits in Ω_0 of which the THEN-conditions are equalities. The IF-conditions of these edits have a non-empty intersection, because both edits are triggered if we fill in the values u_i for the categorical variables in T_1. So, these edits generate an implied edit in Ω_1 if we eliminate variable x_r. The THEN-condition of this implied edit can be written as

$$\sum_{i\neq r} a'_{is} x_i + b'_s = \sum_{i\neq r} a'_{it} x_i + b'_t ,$$

where we have used (9).

Filling in the values u_i for $i \in T_1$ in this implied edit, we find that M_s^E should be equal to M_t^E. In other words, we have constructed an edit in Ω_1 that would be failed if we were to fill in the values u_i for $i \in T_1$. This would contradict our assumption that these values satisfy all edits in Ω_1, and we conclude that two constraints given by (6) (case 1 above) cannot contradict each other.

For cases 2, 3 and 4 we can show in a similar manner that we would be able to construct a failed implied edit in Ω_1. This would contradict our assumption that the values u_i for $i \in T_1$ satisfy all edits in Ω_1, and we conclude that the constraints given by (6) to (8) cannot contradict each other. In turn this allows us to conclude that a value u_r for variable x_r exists such that this value plus the values u_i for $i \in T_1$ satisfy all edits in Ω_0.

Finally, note that if the equality-elimination rule has been applied, using the edit

IF $v_i \in D_i$ (for $i = 1,\ldots,m$) THEN $(x_1,...,x_n) \in \{\mathbf{x} \mid a_{1s}x_1 + ... + a_{ns}x_n + b_s = 0\}$,

to eliminate a continuous variable x_r from the other edits, the value given by

$$x_r = -\frac{1}{a_{rs}}\left(b_s + \sum_{i\neq r} a_{is}u_i\right)$$

plus the values u_i for $i \in T_1$ satisfy the edits in Ω_0. This concludes the proof of the lemma. ■

Theorem 1. If a set of edits corresponding to a particular node is satisfied by the values of the remaining variables in that node, the eliminated continuous variables and the categorical variables that are fixed to a value that differs from their original value together form a (possibly suboptimal) solution to the error localisation problem.

Proof. This follows directly from repeated application of the lemma. ■

In principle, we explore the entire search tree and select the best solutions to the error localisation problem. As the current objective value can only increase while going down the tree, we can prune a branch as soon as the current objective value exceeds the objective value of the best solution found so far. We can also prune a branch if the current objective value exceeds the pre-set threshold $N_{\max}$. Finally, we can prune branches that cannot generate a feasible solution to the error localisation problem because they contain edits that cannot be satisfied, for instance an edit given by "$0 \geq 1$". The solutions found by our algorithm are the optimal ones. This is the content of Theorem 2.

Theorem 2. The algorithm described in this section determines all optimal solutions to the error localisation problem in a mix of categorical and continuous data.

Proof. Theorem 1 provides us with an instrument for determining whether the variables that have been eliminated (in the case of continuous variables) or fixed to their most likely value different from their original value (in the case of categorical variables) to arrive at a particular node together form a solution to the error localisation problem. In principle, the algorithm explores all possibilities of eliminating continuous variables and fixing categorical variables to their most likely value. Note that only one value per equivalence class of a categorical variable needs to be examined. If fixing the categorical variable to this value leads to an optimal solution, fixing the categorical variable to any of the other values in the equivalence class also leads to an optimal solution, and vice versa.

In the algorithm some branches of the search tree are pruned. However, the only branches that are pruned are those that cannot lead to optimal solutions. We conclude that the proposed algorithm performs an exhaustive search for optimal solutions to the error localisation problem. Hence, the algorithm determines all optimal solutions to the error localisation problem. ■

4.2 Selecting the most suspicious variable

The algorithm described in Section 4.1 selects the most suspicious variable in each node. For this selection process one can use a simple heuristic scheme. Such a heuristic scheme could involve the number of violated edits in which the variable is involved (the higher, the more suspicious), the number of satisfied edits in which the variable is involved (the higher, the less suspicious), and the number of violated edits that can be satisfied by changing only the value of the variable under consideration (the higher, the more suspicious). Furthermore, one could make a distinction between numerical equalities and numerical inequalities. For instance, a continuous variable involved in a satisfied equality is less suspicious than a continuous variable involved in a satisfied inequality. Chung (2003) has used these ingredients to develop such a heuristic scheme to find the most suspicious variable.

4.3 Determining the most likely value for a selected categorical variable

In our algorithm, after we have selected a categorical variable in a certain node as the variable that is the most suspicious, we have to determine a value for this variable that is most likely to be correct. In order to do this we fill in all possible values for this categorical variable, or more precisely: we fill in one value from each equivalence class with respect to the current edits. For each such value we estimate the minimal sum of the reliability weights of the variables that still need to be changed. This can be done by taking into account that in each violated edit the value of at least one variable should be changed and also the direction of change of a continuous variable.

In particular, suppose that in a certain node we set the value of the selected categorical variable V to c. We then determine all violated edits given this value and the original values of the remaining variables. In each violated edit we replace a continuous variable x_k by $x_k^0 + x_k^+ - x_k^-$ ($k = 1,\ldots,n$), where x_k^0 denotes the original value of variable x_k, $x_k^+ \geq 0$ a positive change in value, and $x_k^- \geq 0$ a negative change in value (if the original value of x_k is missing we replace x_k by $x_k^+ - x_k^-$). The violated edits are thus transformed into new edit rules involving the new variables x_k^+ and x_k^- as well as the categorical variables. We set the reliability weights of x_k^+ and x_k^- equal to the reliability weight of x_k ($k = 1,\ldots,n$). We then determine the variable r (either a categorical variable, or a positive or negative change to a continuous variable) for which f_r / w_r is the highest, where f_r is the number of times variable r is involved in the violated edits and w_r its reliability weight. If there are several variables for which this fraction is the highest, we choose one of them at random. We assume that all violated edits involving the selected variable can be satisfied. We delete these violated edits and select another variable for the remaining violated edits, using the same selection mechanism. This process goes on until the set of remaining violated edits becomes empty. The sum of the reliability weights of the selected variables is our estimate for the minimal sum of the reliability weights of the variables that still need to be changed given that the selected categorical variable V is set to c. Of all values in the current domain of V we select the value c_0 for which this estimate is the lowest. That value c_0 is the most likely value for variable V. This basic idea can be further elaborated upon by using ideas developed by Chung (2003).

5. Example

In this section we illustrate the idea of the algorithm presented in the previous section by means of an example. This example is similar to an example given by De Waal and Quere (2003). We will not build the entire tree, because this would take too much space and would hardly teach us anything. Instead we will only generate one branch of the tree.

Suppose we have to edit a data set containing four categorical variables v_i ($i = 1,\ldots,4$) and three continuous variables x_k ($k = 1,\ldots,3$). The domain of the first categorical variable is {1,2}, and the domains of the last three categorical variables are {1,2,3}. The set of explicit edits is given by (10) to (21) below.

$$\text{IF } (v_1 = 1 \text{ AND } v_4 \in \{1,3\}) \text{ THEN } \varnothing \tag{10}$$

$$\text{IF } (v_2 = 1 \text{ AND } v_3 = 1) \text{ THEN } \varnothing \tag{11}$$

$$\text{IF } (v_1 = 2 \text{ AND } v_3 \in \{1,3\} \text{ AND } v_4 \in \{1,3\}) \text{ THEN } \varnothing \tag{12}$$

$$x_1 - 12 \geq 0 \tag{13}$$

$$\text{IF } (v_3 \in \{1,3\}) \text{ THEN } x_2 = 0 \tag{14}$$

$$\text{IF } (v_3 = 2) \text{ THEN } x_2 - 1{,}250 \geq 0 \tag{15}$$

$$\text{IF } (v_3 = 2) \text{ THEN } -875 x_1 + 12 x_2 \geq 0 \tag{16}$$

$$\text{IF } (v_3 = 2) \text{ THEN } 1{,}250 x_1 - 8 x_2 \geq 0 \tag{17}$$

$$\text{IF } (v_3 \in \{1,3\}) \text{ THEN } 1{,}250 x_1 - x_3 = 0 \tag{18}$$

$$\text{IF } (v_2 \in \{2,3\} \text{ AND } v_3 = 2) \text{ THEN } 1{,}250 x_1 + 12 x_2 - x_3 + 1250 = 0 \tag{19}$$

$$\text{IF } (v_2 = 1 \text{ AND } v_3 = 2) \text{ THEN } 1{,}250 x_1 + 12 x_2 - x_3 = 0 \tag{20}$$

$$x_3 - 20{,}000 \geq 0 \tag{21}$$

If a categorical variable is not mentioned in an IF-condition, this variable may take any value in its domain. For instance, edit (10) actually means

$$\text{IF } (v_1 = 1 \text{ AND } v_2 \in D_2 \text{ AND } v_3 \in D_3 \text{ AND } v_4 \in \{1,3\}) \text{ THEN } \varnothing,$$

where D_i is the domain of categorical variable i.

Now, suppose that a record with values $v_1 = 1$, $v_2 = 2$, $v_3 = 2$, $v_4 = 1$, $x_1 = 25$, $x_2 = 3{,}050$ and $x_3 = 90{,}000$ is to be edited. Edits (10) and (19) are failed, so this record is inconsistent. The reliability weight of each variable equals 1. We apply the algorithm described in the Section 4 and start by selecting the variable that is the most suspicious. In this example, we simply compute the number of times that each variable is involved in a satisfied edit and the number of times that each variable is involved in a violated edit. All variables are involved in one violated edit. Variables v_1, v_2 and v_4 are involved in one satisfied edit, the other variables are involved in two or more satisfied edits. We therefore consider variables v_1, v_2 and v_4 to be the most suspicious ones. We select one of them randomly, say we select variable v_1. The most likely value for this variable is 2, in which case the estimated number of variables that still need to be changed equals 1. The equivalence class of 2 with respect to the current edits is {2}. We now construct two branches: in one branch the value of v_1 is set to 2, in the other branch the domain of v_1 is updated to D_1 = {1}. In this paper we only consider the former branch and set v_1 equal to 2. We hence fill in $v_1 = 2$ in the edits, delete variable v_1 from the appropriate edits, and obtain the following current set of edits: (11), (13) to (21), and

$$\text{IF } (v_3 \in \{1,3\} \text{ AND } v_4 \in \{1,3\}) \text{ THEN } \varnothing, \tag{22}$$

Edit (22) is obtained from (12). We select the variable that is now considered the most suspicious, say we select variable v_2. We again construct two branches: one where v_2 is set to its most likely value (1) and another where the domain of v_2 is updated to {2,3}. We consider only the latter branch. The edits that cannot be triggered by this domain are deleted. In the new node of the tree, the domain of v_2 forms a single equivalence class. We hence delete variable v_2 from the edits, and obtain (13) to (18), (21), (22), and

$$\text{IF } (v_3 = 2) \text{ THEN } 1{,}250x_1 + 12x_2 - x_3 + 1{,}250 = 0,$$

which is obtained from (19). We once again select the variable that is considered the most suspicious, say variable x_3. We construct two branches: one where x_3 is eliminated from the set of edits and another where x_3 is fixed to its original value. We consider only the former branch and eliminate x_3 from the set of edits. We obtain the following edits: (13) to (17), (22),

$$\text{IF } (v_3 \in \{1,3\}) \text{ THEN } 1250x_1 - 20{,}000 \geq 0 \tag{23}$$

and

$$\text{IF } (v_3 = 2) \text{ THEN } 1{,}250x_1 + 12x_2 - 18{,}750 \geq 0.$$

For instance, edit (23) is obtained by combining edits (18) and (21). All current edits are satisfied if we fill in the original values of the remaining variables. This implies that the set of original, explicit edits can be satisfied by changing the values of x_3 and $v_{1,}$ and fixing the other variables to their original values. In other words, a solution to the error localisation problem for this record is given by: change the values of x_3 and v_1. Possible values, the only ones in this case, are $v_1 = 2$ and $x_3 = 69{,}100$. It is easy to check that the resulting record indeed satisfies all explicit edits.

The other branches of the search tree, which we have skipped, also need to be examined, because it is possible that they contain a better solution to the error localisation problem. By exploring all branches of the search tree one can arrive at all optimal solutions to the error localisation problem for the record under consideration.

6. Extension to integer-valued data

If some of the variables are integer-valued, special measures must be taken to ensure that these variables can indeed attain integer values. For a mix of categorical, continuous and integer data, we first treat the integer-valued variables as being continuous. That is, first we find a, possibly suboptimal, solution to the error localisation problem where all numerical variables are being treated as continuous ones. We refer to such a solution as a *continuous* solution. Given a continuous solution, we check whether the integer-valued variables involved in this solution can attain integer values. This is obviously the case if no integer-valued variables are involved in this continuous solution. If integer-valued variables are involved in the continuous solution, we start by filling in the original values of the variables not involved in this solution and the selected values for the categorical variables involved in this solution into the edits. This leads to a system of linear (in)equalities involving only the numerical variables in the continuous solution. Subsequently, we eliminate the continuous variables by means of Fourier-Motzkin elimination (see Section 3). We then obtain a system of linear (in)equalities involving only integer variables. The continuous solution to the error localisation problem is obviously an actual solution to the error localisation problem if and only if there is a feasible integer solution to this system of linear (in)equalities involving only integer variables. We can check whether this is indeed the case by applying the so-called Omega test. Testing whether or not the integer-valued variables in a continuous solution can attain integer values is a so-called NP-complete problem. In worst-case instances, the Omega test therefore requires an enormous amount of computing time. However, such worst-case instances seem to occur rarely in practice. For his application of the Omega test, Pugh (1992) claims a good performance. We refer to Pugh (1992) and De Waal (2003) for details about the Omega test.

7. A heuristic based on the branching scheme

If a record contains many errors, the exact method described so far may be too time-consuming to use. In such a case one can resort to applying a heuristic that yields a suboptimal solution rather than all optimal ones. In this section we propose a heuristic based on the exact algorithm proposed earlier in this paper. This heuristic is suitable for a mix of categorical, continuous and integer data.

1) Variable selection: Select the variable of which the value should most likely be changed. If the selected variable is continuous, go to Step 2a. If the selected variable is integer-valued, go to Step 2b. If the selected variable is categorical, go to Step 2c.

2) Treating the selected variable:

   a) Eliminate the selected continuous variable from the current set of edits using the elimination technique of Section 3.

   b) We denote the selected integer variable by x_r. We now consider two cases.

      i) If x_r enters an edit of type (5), we try to rescale the coefficients of the equality in such an edit so that the coefficient of x_r equals 1 or -1, and the coefficients of the other variables remain integer. If such an equality occurs in an edit of type (5) involving x_r, we use its rescaled version to eliminate x_r from the current set of edits by means of the equality-elimination rule of Section 3.

If none of the equalities in the edits of type (5) involving x_r can be rescaled in this manner, we choose an edit of type (5) involving x_r at random and we output the message that the solution that will be generated by the heuristic may possibly not be valid because some integer-valued variables involved in the solution may not have feasible integer values. We use the chosen edit to eliminate x_r from the current set of edits by means of the equality-elimination rule of Section 3.

ii) If x_r does not enter an edit of type (5), we consider all edits in format (1) involving x_r pair-wise. Suppose we consider the pair consisting of edit s and edit t. We first check whether the intersection of the IF-conditions of these edits is non-empty, i.e. whether the intersections $F_i^s \cap F_i^t$ are non-empty for all $i = 1,\dots,m$. If any of these intersections is empty, we do not have to consider this particular combination of edits anymore. So, suppose that all intersections are non-empty. Now, we again consider two cases.

(1) We first consider the case that at least one of the THEN-conditions of edits s and t is an equality. If the THEN-condition of edit s is an equality, we try to rescale the coefficients in the THEN-condition of edit s so that the coefficient of x_r equals 1 or -1, and the coefficients of the other variables remain integer. If this is possible, we use the rescaled THEN-condition of edit s to eliminate x_r from edits s and t by means of the technique of Section 3. If the THEN-condition of edit s is not an equality or cannot be rescaled in this manner, we examine whether edit t possesses the desired property. If so, we use the rescaled THEN-condition of edit t to eliminate x_r from edits s and t by means of the technique of Section 3. If neither of the edits possesses the desired property, we apply the technique of Section 3 to eliminate x_r from edits s and t, and we output the message that the solution that will be generated by the heuristic may possibly not be valid because some integer-valued variables involved in the solution may not have feasible integer values.

(2) We now consider the case that the THEN-conditions of both edits are inequalities. We check whether $a_{rs} \times a_{rt} < 0$. If that is not the case, we do not consider this particular combination of edits anymore. If $a_{rs} \times a_{rt} < 0$, we compute

$$\tilde{b} = |a_{rs}| \times b_t + |a_{rt}| \times b_s - (|a_{rs}| - 1) \times (|a_{rt}| - 1)\,. \qquad (24)$$

This yields an edit for the new set of edits. The IF-condition of this new edit is given by $F_i^s \cap F_i^t$ for $i = 1,\dots,m$, the THEN-condition is defined by (3), (4) and (24). Together (3), (4) and (24) form the so-called dark shadow of the edits s and t (see Pugh, 1992).

After all new edits have been generated, all edits involving x_r are deleted. In this way we obtain a new set of edits not involving x_r.

c) For each value in the domain of the selected categorical variable, we estimate the sum of the reliability weights of the variables that still need to be changed. We choose the value in the domain of the selected variable for which this estimate is minimal. The new set of edits is obtained by filling in this value for the selected variable into the edits. Edits that cannot be triggered anymore are removed from the set of edits. The selected categorical variable is subsequently deleted.

After the completion of Steps 2a, 2b, or 2c: if the resulting set of edits is satisfied by the values of the remaining variables, we have found a, possibly suboptimal, solution to the error localisation, and stop. If the resulting set of edits is violated by the values of the remaining variables, we go to Step 1. ■

For Steps 1 and 2c, the selection of a variable and the selection of a value for a categorical variable, we can use the heuristics sketched in Sections 4.2 and 4.3

8. Discussion

In this paper we have proposed a simple branching scheme for solving the error localisation problem in a mix of categorical and continuous data to optimality. We have also proposed an extension to integer-valued data based on the so-called Omega test (see Pugh, 1992). Furthermore, we have proposed a heuristic for finding suboptimal solutions to the error localisation problem, which is based on the exact algorithm.

The performance of the proposed algorithm can possibly be improved upon in a number of ways, for instance by

- pre-processing the set of edits before running the algorithm;
- computing lower bounds on the objective value for the current branch;
- detecting inconsistency during tree generation by more advanced techniques than standard constraint propagation techniques;
- detecting redundant constraints during tree generation.

For more information on such improvements we refer to De Waal (2001).

Several research topics remain to be examined. First of all, the computing time of the exact algorithm needs to be evaluated and compared to the computing times of other algorithms for the error localisation problem. Second, in the exact algorithm that we propose in the present paper, we select the most likely value for a categorical variable that is being treated. We hereby aim to construct a good solution to the error localisation problem as soon as possible. Such a solution can then be used to prune the search tree. Alternatively, one could consider selecting the most *unlikely* value for the categorical variable under consideration. The goal of this approach is to detect infeasible branches soon, and thereby prune the tree. It remains to be examined which of the two alternatives leads to the best computational results. Finally, the heuristic algorithm needs to be evaluated in terms of the required computing time as well as the quality of the solutions determined.

References

Chandru, V. and J. N. Hooker (1999), *Optimization Methods for Logical Inference*. John Wiley & Sons, New York.

Chung, W.H. (2003), *Effective Automatic Error Localisation* (in Dutch). Internal report (BPA number: 1057-03-TMO), Statistics Netherlands, Voorburg.

De Waal, T. (2001), *Potential Improvements in Leo/Cherry Pie and ECS*. Report (research paper 0116), Statistics Netherlands, Voorburg.

De Waal, T. (2003), *Processing of Erroneous and Unsafe Data*. Ph.D. Thesis, Erasmus University, Rotterdam.

De Waal, T. and R. Quere (2003), A Fast and Simple Algorithm for Automatic Editing of Mixed Data. Submitted to *Journal of Official Statistics*.

Duffin, R.J. (1974), On Fourier's Analysis of Linear Inequality Systems. *Mathematical Programming Studies 1*, pp. 71-95.

Fellegi, I.P. and D. Holt (1976), A Systematic Approach to Automatic Edit and Imputation. *Journal of the American Statistical Association 71*, pp. 17-35.

Hooker, J. (2000), *Logic-Based Methods for Optimization: Combining Optimization and Constraint Satisfaction*. John Wiley & Sons, New York.

Pugh, W. (1992), The Omega Test: A Fast and Practical Integer Programming Algorithm for Data Dependence Analysis. *Communications of the ACM 35*, pp. 102-114.