Table of Contents.

I Define the meaning of a table

We shall use the following symbols:

i = row number within condition area
k = row number within action area
I = no. of condition rows
K = no. of action rows
j = rule number
J = no. of rules (excluding the ELSE rule)
c_{ij} = contents of cell (i,j) in the condition area
a_{kj} = contents of cell (k,j) in the action area
v_i = contents of stub-cell in condition row i

The following assumptions are made about the 'meaning' of a table:

1. A table with two or more rules that could be satisfied simultaneously is invalid.
2. Hence rules may be freely interchanged by the processor.
3. Within the condition area, the order in which the conditions are tested is immaterial; hence condition-rows may be freely interchanged.
4. The actions for any rule are to be performed in the order indicated in the table.

5. The various conditions listed in any one row are not necessarily either exclusive or exhaustive

II. Major Characteristics of the Condition Area.

i. Relation-ships between conditions in a single row.

(a) Exclusive, exhaustive set

A set of conditions $c_1\ c_2 \cdots c_n$ forms an exclusive, exhaustive set if one and only one of the conditions in the set is true (at any given time).

Examples: $\alpha = \beta$; $\alpha \neq \beta$

$\alpha < \beta$, $\alpha \geq \beta$

$\alpha < \beta$, $\alpha = \beta$, $\alpha > \beta$

α IS +VE , α IS −VE , α IS ZERO

α IS LISTA , α NOT LISTA

In such a set, the truth of one condition implies the falsity of all the others, and the falsity of all but one of the conditions implies the truth of the remaining one

(b) Exclusive, non-exhaustive set.

None or one of the conditions is true (at any given time).

Examples. $\alpha = 1$, $\alpha = 2$, $\alpha = 3$, $\alpha = 4$

$\alpha < \beta$, $\alpha > \beta$

α IS +VE , α IS ZERO

In such a set, the truth of one condition implies the falsity of all the others

(c) <u>Order-related set.</u>

A set of conditions $c_1, c_2, \ldots, c_n$ is order related if ~~th~~ either (a) the truth of any one condition c_i implies the truth of conditions $c_{i+1}, c_{i+2}, \ldots, c_n$ or (b) the falsity of c_i implies the falsity of $c_{i+1}, c_{i+2}, \ldots, c_n$.

Examples. $\alpha \le 25$, $\alpha \le 30$, $\alpha \le 40$
$\alpha \le 5$, $\alpha \le 4$, $\alpha \le 3$

(d) <u>Unrelated conditions.</u>

If it is not possible by examination of of a table (without knowledge of the data definitions) to detect any of the above relations, we assume that ~~[illegible]~~ the conditions in a row are unrelated.

Thus for

$$\alpha = \beta, \quad \alpha = \gamma, \quad \alpha = \delta$$

we must assume that the set is neither exhaustive nor exclusive, ~~The table user may assume that the set is exclusive or exhaustive or both~~, ~~but~~ except that where a table would be invalid unless a set of apparently unrelated conditions were in fact an exclusive set, the processor may legitimately assume that the conditions are ~~exter~~ exclusive (see para (e) below)

(e) "Apparent" exclusivity

It is evident that in a table such as

α	-	0	= 0
β	γ	δ	= ϵ

the table is invalid unless the conditions in the second row form an exclusive set. Further, we have posited in our basic assumptions the complete interchangeability of rules, so that if, for example, rules 1 and 3 are both satisfied simultaneously (because $\beta = \gamma = \epsilon$) there is no reason to choose rule 1 as the effective solution rather than rule 3. Therefore it is legitimate in the present case to treat the conditions of row 2 as an exclusive set, since the existing ambiguity is not increased by this assumption.

We distinguish between the set "$\alpha = 2$, $\alpha = 3$, $\alpha = 4$" and ~~the~~ sets such as the one above by calling the former a <u>necessarily exclusive set</u> and the latter an <u>apparently exclusive set</u>

(f) Symbolic representation of conditions

For greater ease and legibility, we can represent the conditions in any given row by numbers, as follows:

(i) Each different unrelated condition is represented by a different number. Numbers are assigned serially, 1, 2, 3, ..., reading from left to right.

(ii) An exclusive non-exhaustive set of conditions (necessarily or apparently exclusive) is indicated by a letter e following the numbers.

(iii) An ordered set is indicated by a letter h following the condition numbers.

(iv) An exhaustive, exclusive set of conditions are represented by a single condition number, since a single test operation will determine the state (true or false) of all conditions in the set. Thus a dichotomy is represented by the symbols 1a and 1b, a trichotomy by 1a, 1b, and 1c.

(v) Empty ("non-pertinent") cells are represented by an X.

~~See~~

Thus, for example:

$\alpha = 0$	$\alpha = 0$	$\alpha = 0$	$\alpha = 0$	$\alpha \neq 0$	$\alpha \neq 0$	$\alpha \neq 0$
$\beta < 10$	$\beta < 10$	$\beta = 10$	$\beta > 10$	$\beta < 10$	$\beta = 10$	$\beta > 10$
$\gamma = \mu$	$\gamma = \rho$	x	x	$\gamma = 1$	$\gamma = 2$	$\gamma = 5$

becomes

1a	1a	1a	1a	1b	1b	1b
1a	1a	1b	1c	1a*	1b*	1c*
1a	2e	x	x	3c	4c	5c

2 Emptiness

(a) Minimum emptiness is zero

(b) Maximum emptiness is limited by the fact that each row and each column must have at least 1 non-empty cell. Also, for the table to be valid, each rule must be unique, so that no pair of rules may be compatible in each row (2 cells are compatible if both cells or either cell is empty.) Thus a table of maximum emptiness would look like this:

1	2	3
1		
	1	
		1

There must be at least 2J non-empty cells (unless the table has only one row, in which case the emptiness must be zero). If we define emptiness as number of empty cells divided by total no. of cells, we have

$$\text{maximum emptiness} = (I-2)/I.$$

3. Degrees of freedom.

Let d_i represent [illegible] of freedom of [illegible]

If the conditions in row i form a dichotomy or a trichotomy, d_i is 2 or 3 respectively.

Otherwise, d_i equals number of different conditions listed in row i, plus "other".

E.g. in the table on page II. 5

$$d_1 = 2$$
$$d_2 = 3$$
$$d_3 = 6$$

The number of possible combinations of different values of the v_i is $\prod d_i = 6 \times 3 \times 2 = 36$

Define column-weight w_i as the number of possible combinations covered by a column. For a column with no X's, the weight is 1. For a column containing X's, the weight is the product of the d for those cells which have X's in them.

If the rules shown in the table do not cover all the possible combinations (i.e. sum of the column-weights is less than $\prod d_i$), the balance are covered by the ELSE column (treating the error-table as an ELSE column when a table has no explicit ELSE) Thus there is a weight , w_E , attached to the error column.

Define a *perfect table* as one in which the rules cover all the possibilities , so that $w_E = 0$.

4. Repetition

a) Definition

A 'repeat' condition cell is one that is identical with some cell further to the left in the same row.

b) A table with no repeats is possible. For example:

CODE =	01	02	03	04	05
PART IS	LISTA	LISTB	LISTC	LISTD	LISTE

In such tables, only one of the condition-rows has a logical function (ie determines what to do next). The other rows are essentially edit functions. Thus in the table above, row 2 is an edit check upon PART, the particular check depending upon the value of CODE.

c) Tables with no repeats are of little interest to the present discussion, since the best algorithm is a straightforward rule-by-rule approach (first arranging the rules in descending order of probability.

d) Maximum repetitiveness occurs in a perfect limited-entry table with zero emptiness, such as:

V_1	Y	Y	Y	N	N	N	N	Y
V_2	Y	N	N	Y	Y	N	N	Y
V_3	N	Y	N	Y	N	Y	N	Y

The maximum no. of repeats possible for any given number of condition rows is shown below.

No. of rows	Maximum no. of rules	Max. no of cells	Unique cells	Max no of repeats
2	4	8	4	4
3	[illegible]	24	6	[illegible]
4	16	64	8	[illegible]
5	32	160	10	150.

e) In general, if

I = no of rows when table is reduced to limited-entry form *.

J = no. of rules.

E = no of empty cells

$$\text{No. of repeats} = I.J - 2I - E$$

* Assuming there are no 'edit' rows with the same condition appearing in every rule.

5. Relationships between conditions in different rows

A) Lemma 1. Any [illegible] row can be [illegible] a series of limited-entry rows without altering the meaning of the table.

Eg.

α	$=1$	$=1$	$=2$	$=2$
β	<10	≥ 10	<15	≥ 15

can be expressed as

$\alpha = 1$	Y	Y		
$\alpha = 2$			Y	Y
$\beta < 10$	Y			
$\beta \geq 10$		Y		
$\beta < 15$			Y	
$\beta \geq 15$				Y

B) Lemma 2. If any two condition-rows have a common stub variable, and if there is no rule having non-empty cells in each of the two rows, then the rows can be combined to form a single row.

c) In examining a table to locate any related sets of conditions, it is not enough to scan within each row, there may also be relations between conditions in different rows. To simplify the table examination, it is probably best to arrange that at read-in time either (a) all extended entry rows will be reduced to limited-entry ~~ext~~ or (b) all combine-able rows will be combined. The latter choice seems preferable, since the result is more compact.

6 Checking for Logical Correctness

(a) To check a table for logical correctness, compare each possible pair of rules for mutual exclusivity.

(b) To check that two rules, r_1 and r_2, are mutually exclusive, compare corresponding elements in each rule.

(i) Throw out a pair of elements if either one, or both, is an X

(ii) Throw out a pair of elements if they are identical

(iii) There must remain at least one pair of elements which form an exclusive set. If not, r_1 and r_2 are not mutually exclusive.

(c) It might be helpful to allow the table-writer to indicate when the conditions in a row, though not evidently exclusive, are in fact to be assumed to be exclusive. This would avoid "error messages" about tables which appear invalid to the processor, but which the user knows to be valid.

III. Condition-Area Algorithms.

1. Pre-testing algorithm.

a) Description.

"Pre-testing" means that all the different tests are performed ahead of time, and indicator-bits set showing the results, one bit per unique condition (note: a dichotomous pair of conditions requires 2 bits, not 1). The resulting status-word is compared to mask-words to determine which rule succeeds. There is one mask-word per rule, generated at compile-time.

b) Space rating

This method is economical of space; no test is stored more than once.

c) Time rating

Instead of masks one could use a unique number for each rule which would be generated by pre-testing. Use unique power of 2 for each condition.

The method is obviously poor on any computer which does not have a fast 'mask' instruction (eg the 7080)

It is best on a computer such as 7090, with the ability to 'mask' a 36-bit word in one instruction. Even here, the method beats path-[illegible] only if N the number of different tests in the table, is less than the average path-length under path-optimisation. (See the formulae below)

d) Timing formulae

Let

N = no. of different tests in the table

t_{avg} = avg amount of time to make a test

p_j = probability that rule j is the solution

p_E = - - - - - - rule E (error or else) is solution

n_j = no of tests (on average) in the path leading to rule j

t_{cmp} = time required to compare the status-word to a mask-word (ie program loop time)

τ = average table solution time

Then using the pre-testing method

$$\tau = N \cdot t_{avg} + t_{cmp} \left(p_1 + 2p_2 + \cdots + J p_J + J p_E\right. \qquad \text{(III.1)}$$

Using a path-optimisation method

$$\tau = p_1 n_1 t_{avg} + p_2 n_2 t_{avg} + \cdots \qquad \text{(III.2)}$$

$$\cdots + p_J n_J t_{avg} + p_E n_E t_{avg}$$

(a) <u>The table as flow-chart.</u>

A path-optimisation algorithm is essentially a set of rules for converting the condition-area of a table into a "flow-chart". However once rules have been defined for tracing failure-paths through a table, the table itself becomes an explicit flow-chart. Since the tabular form is more compact and easier to write than free-form flow-charts, we shall consider the problem of developing an optimal flow-chart from a table to be essentially the problem of finding the optimal order of rows and columns. (Note, however, that row order may vary from one group of columns to another)

(b) <u>Rules for path-tracing</u>

~~We shall assume that~~

(i) <u>Basic</u>

The basic rules for path-tracing are

— if c_{ij} succeeds, go to $c_{i+1,j}$

— if c_{ij} fails, go to $c_{i,j+1}$

(ii) <u>First improvement.</u>

The first level of improvement is that employed in the 7090 algorithm, namely

— any c_{ij} which is a repeat of the cell to its left and which has only repeat cells above it may be suppressed (indicated by circling the condition-symbol)

- The failure-path from any cell (i, j) skips over suppressed cells + goes to the first unsuppressed cell encountered in a scan from row 1 through row i in columns $j+1$, $j+2$, ….
- suppressed cells are dropped ie they generate no object coding.

(iii) <u>Second-level improvement.</u>

The second level of improvement is to take advantage of any exclusive, exhaustive sets of conditions, to predict success of any member of such a set when it is known that all other members have failed (along the path currently being traced)

Thus if cell (i, j) has a failure-path ~~to~~ to cell $(i' j')$, and (i, j) and $(i' j')$ are a dichotomous pair of conditions, then the improved failure-path from $(i j)$ goes directly to $(i'+1, j')$

This improvement is relatively easy to implement for a dichotomy. For a trichotomy it is somewhat harder.

(iv) <u>Third-level improvement.</u>

The third level of improvement is to take advantage of exclusive sets of conditions (exhaustive or not) to predict failure of any given member of such a set when it is known that ~~on~~ some other

[illegible]

Thus if cell (i, j) has a failure-destination at (i', j'), and $i' < i$, we examine the cells in row i' through row $i+1$ of column j'. If we find that ~~once~~ one of these cells (i'', j') is a member of an exclusive set, and that the most proximate unsuppressed cell to its left is another member of that set, then the true failure-destination from cell (i, j) is the failure-destination (yet to be determined) of (i'', j')

(v) <u>By-passed cells</u>

The second- and third-level improvements may result in certain cells' being completely by-passed. Such cells generate no object-coding.

(c) <u>Evaluating a path-optimisation algorithm.</u>

The better of two alternative algorithms is the one that gives the smaller τ, where

$$\tau = p_1 t_1 + p_2 t_2 + \cdots + p_J t_J + p_E t_E$$

t_j = avg. time to determine that solution is rule j.
= avg. path-length to rule j × avg. test-time.

In evaluating any algorithm, we shall express the algorithm in terms of rules for row and column

... apply the path-length rules given above in order to compute path-lengths and hence $\bar{L}$. We assume that all levels of improvement outlined above will apply.

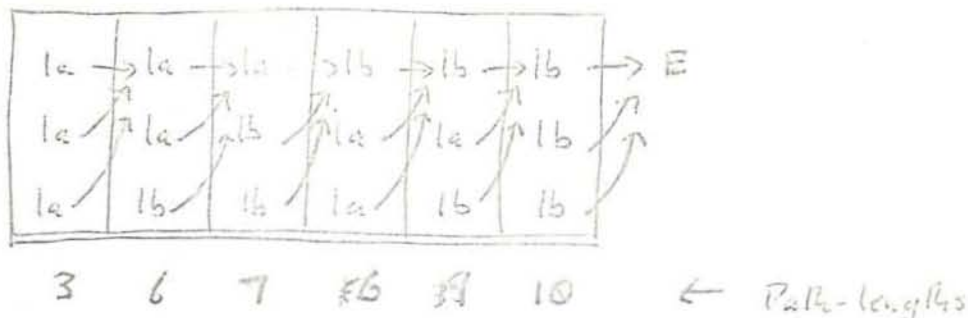
Note that in some cases there may be more than one possible route to rule j. Lacking any information on the relative probabilities of the possible routes (and it seems likely that one would lack this data), the average path-length over all possible routes must be computed.

(d) Symbols.

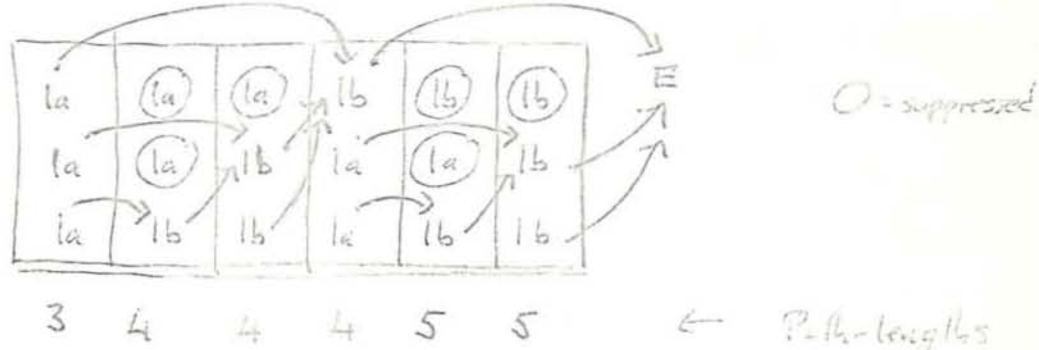
The following symbols will be used in representing the condition matrix:

1, 2, 3, . . . : a set of unrelated conditions
1e, 2e, 3e, . . . : a set of exclusive conditions
1A, 1B : a dichotomous pair of conditions
1A, 1B, 1C : a trichotomous set of conditions
1 : a suppressed or bypassed cell
* : a suppressed cell (an adjacent redundancy)

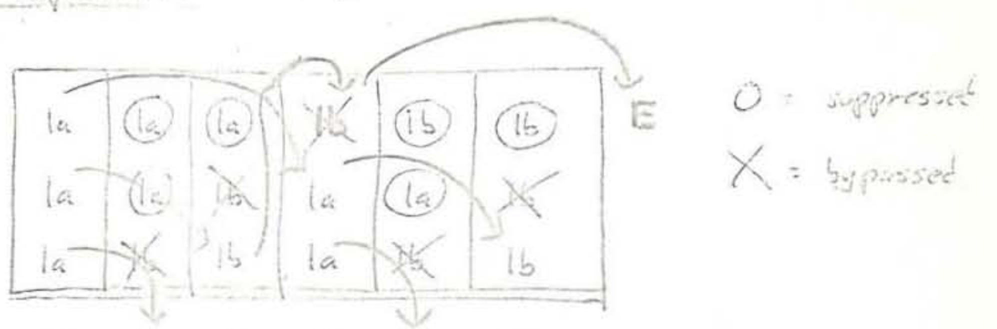
Basic level 18



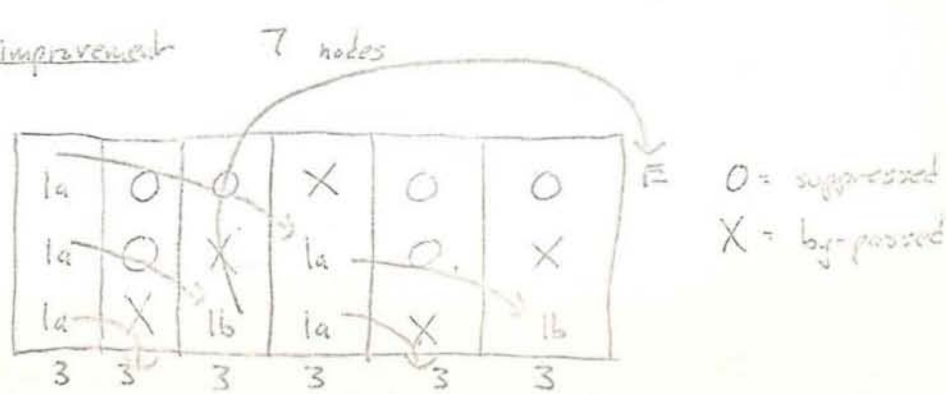
First level improvement 12 nodes



Second-level improvement 8 nodes



Third-level improvement 7 nodes



(a) Introduction

This section will attempt to define an algorithm for shortening the average path-length, assuming that all rules are equally probable and that the E rule has a probability so low that it can be ignored.

(b) Choice of initial test.

As the initial test (ie the top row, left side) we would prefer:

(i) a single condition that spans all rules

(ii) a dichotomous or trichotomous set of conditions that spans all rules.

Failing such a test, we are forced to choose a test which will be irrelevant to one or more rules. For example:

	I	II	III	IV	V	VI	VII
A	1e	1e	1e	2e	2e	3e	3e
B	1	1	1	1	X	1	1
C	1a	1a	1b	1b	X	1a	1b
D	1	2	X	1	2	X	X

If we start with row A, any one test in that row is relevant to only a few rules, & we have wasted time if the solution lies in a different rule. If we start with rows B or C, we waste time only if the solution is rule V

[illegible]
possible arrangements, with path-lengths shown:

	I	II	III	IV	V	VI	VII	
A	1e	*	*	2e	*	3e	*	
B	1	*	*	1	X	1	*	
C	1a	*	1b	1b	X	1a	1b	
D	1	2	X	1	2	X	X	
	4	5	3	5	{4, 5, 6}	5	5	Total - 32

	I	II	III	VI	VII	IV	V	
B	1	*	*	*	*	*	X	
A	1e	*	*	3e	*	2e	*	
C	1a	*	1b	1a	1b	1b	X	
D	1	2	X	X	X	1	2	
	4	5	3	4	4	6	{3, 6, 7}	Total = $31\frac{1}{3}$

	I	II	VI	III	IV	VII	V	
C	1a	*	*	1b	*	*	X	
B	1a	*	2e	1	*	*	X	
A	1e	*	3e	1e	2e	3e	2e	
D	1	2	X	X	1	X	2	
	4	5	4	3	5	5	{4, 4, 6, 6}	Total = 31

B	1	x	x	x	x	x	X
C	1a	x	x	1b	x	x	X
A	1e	x	3c	1e	2e	3e	2c
D	1	2	X	X	1	X	2
	4	5	4	3	5	5	$\left\{\begin{matrix}3\\5\\6\end{matrix}\right\}$

Total 30 ⅔.

(c) A possible [illegible] algorithm

(i) Find the most relevant test (ie the most repeated condition, counting 1b + 1c as repeats of 1a) in the matrix. Of two equally relevant tests, prefer a single-valued one to a 2- or 3-valued one. Of two equally relevant single-valued tests, prefer the row with the largest net number of repeats.

(ii) Bring the row containing this most-repeated test to the top, + re-arrange the rules to bring repeated conditions adjacent, (most-repeated left-most, least-repeated ~~left~~ right-most), and X's to the right.

(iii) Examine the remaining rows to find the most-repeated test but do not count a repeat that crosses the boundary between 'strips'. A strip is a consecutive group of rules having a repeated condition in row 1.

rules *within* each strip.

(v) Re-partition the matrix into strips, taking row 1 and row 2 into account. Repeat (iii) and (iv) to get the 3rd row into place.

(vi) Continue this until the remaining row(s) have no 'countable' repeats. If there are two or more remaining rows, bring rows with exclusive condition-sets above rows of unrelated conditions.

(vii) The resulting re-arrangement is the final form of the matrix. Now trace the failure paths as described above, & then code rule by rule.

(d) Example 1

A highly 'ordered' original table such as

	I	II	III	IV	V
A	1a	1a	1a	1a	1b
B	1a	1b	1b	1b	X
C	X	1a	1b	1b	X
D	X	X	1a	1b	X
E	X	X	X	1a	X

	III	IV	II	I	V
A	1a	x	x	x	1b
B	1b	x	x	1a	X
C	1b	x	1a	X	X
D	1a	1b	X	X	X
E	X	1b	X	X	X
	4	5	3	2	1

Path lengths

The re-arrangement is not quite as easy to read as the original, but it has the same minimal path lengths.

(e) <u>Example 2.</u>

	I	II	III	IV	V	VI	VII	VIII	IX	X
A	1e	1e	1e	2e	2e	2e	3e	3e	4e	X
B	1e	1e	2e	1e	1e	2e	1e	2e	X	3e
C	1e	2e	X	X	X	X	2e	1e	X	X
D	X	X	X	1e	2e	X	X	X	1e	2e
	3	4	3	4	5	4	5	6	5	↖ 5 (min)

becomes

	I	II	IV	V	VII	III	VI	VIII	IX	X
B	1e	1e	1e	1e	1e	2e	2e	2e	3e	X
A	1e	1e	2e	2e	3e	1e	2e	3e	X	4e
C	1e	2e	X	X	2e	X	X	1e	X	X
D	X	X	1e	2e	X	X	X	X	2e	1e
	3	4	4	5	5	3	4	6	4	↖ 5 (min) to 8 (reasonable max

Avg = 4.25.

(i) Let us call 'repeats', as defined above, redundant cells. Consider now the point when all rows containing redundant cells have been moved to the top portion of the matrix. Either these rows have already uniquely identified each rule (ie no 2 rules can simultaneously succeed in all rows from row 1 to row n), or they have not. If they have not, there must remain, for each set of ambiguous rules, a set of exclusive conditions. Eg if the matrix so far is

1	2	3	4	5	6
1a	1a	1a	1b	1b	1b
1a	1a	1b	1a	1a	1b

There must remain a set of conditions to distinguish between rules 1 + 2, and a set to distinguish between rules 4 + 5. These sets may or may not be in the same row.

(ii) Our algorithm therefore states that after all redundant rows have been moved to the top portion of the table, any row(s) containing exclusive sets of conditions should be placed next. Thus for example the table shown above might become :-

1a	1a	1a	1b	1b	1b
1a	1a	1b	1a	1a	1b
1e	2e	X	X	X	X
X	X	X	1e	2e	X

(iii) Let us call the rows necessary to identify each rule uniquely the <u>discriminatory</u> portion of the re-arranged condition matrix. Any remaining rows constitute the <u>edit</u> portion. The edit portion always lies below the discriminatory portion.

(iv) Clearly, if a test in the edit portion fails, the failure-path from this test will go directly to rule E (~~since~~ no two rules can succeed simultaneously in all rows of the discriminatory portion, ~~if one~~ therefore if one succeeds all others <u>must</u> fail & there is no need to pass through them) Therefore there are no paths from one rule to another within the edit portion. Therefore it does not matter how we order the rows in the edit portion, since the order has no effect on path lengths.

(v) <u>Example</u> (see next page)

1e	1e	1e	2e	2e	2e	discriminatory portion
1a	1a	1b	1a	1a	1b	
1e	2e	x	x	x	x	
x	x	x	1e	2e	x	
1	x	2	1	x	2	edit portion
1	x	x	x	1	x	
5	4	3	5	6	4	← path-lengths

The "path-lengths" remain unchanged if we reverse the two rows of the edit portion in the table above.

1e	1e	1e	2e	2e	2e	discriminatory portion
1a	1a	1b	1a	1a	1b	
1e			x	x	x	
x	x	x	1e	2e	x	
1	x	x	x	1	x	edit portion
1	x	2	1	x	2	
5	4	3	5	6	4	