A Description and User's Guide for CLUSTER/2
A Program for Conjunctive Conceptual Clustering

Robert Stepp

University of Illinois
Department of Computer Science
Intelligent Systems Group

ABSTRACT

This report presents a comprehensive reference for using the program
CLUSTER/2 for constructing classifications of arbitrary objects. Specifically, given a
set of object descriptions (in the form of attribute-value pairs) the program structures
the objects into a hierarchy of classes. Each class is characterized by a conjunctive
description involving selected object attributes. Object descriptions and control
parameters are presented to the program in the form of relational tables. This document
contains the definitions of the relational table syntax and the values that can be
entered. Sample input and the corresponding program output generated by
CLUSTER/2 are discussed.
1. Introduction

CLUSTER/2 is a program for automatically constructing conceptual classifications. It is the successor to the program CLUSTER/PAF [1],[2]. CLUSTER/2 may be applied to a wide variety of problem domains. Experimentation with the program has included varied practical problems such as classifying Spanish folk songs [3], classifying microcomputers [4], and reconstructing soybean disease categories [4]. Input to CLUSTER/2 consists of a set of attribute vectors, (one vector per input event), definitions of the types of variables and their value sets, and a specification of the clustering quality criterion. The program generates a classification hierarchy of the data events along with a conjunctive description of each cluster at each level of the hierarchy. The program forms clusters that have a conjunctive description and optimize the given criterion of clustering quality. The program forms both clusters and their descriptions, and thus behaves quite unlike programs that implement conventional numerical taxonomy techniques. The theoretical background for CLUSTER/2 can be found in [4] and [5].

2. Input file content

CLUSTER/2 is run in batch mode. All parameters are provided from a single input file. The program creates one output file which is a report of the obtained clusters along with optional trace information which shows what alternative clusterings were considered while obtaining the final reported solution.

The input data are in the form of relational tables according to a standard syntax. A table consists of three parts: a table-identifier that is composed of one of the predefined table names, a line of column-headings (composed of predefined column names), and lines of data values arranged such that the order of the data on each line matches the order of the column headings. Rather than having a prescribed order, the order of the columns is determined by the order of the column-heading names. In the example tables that follow, the columns are arranged in a convenient order. The user of CLUSTER/2 may freely permute the order of the columns in any table.

Data items in one line of a table are separated by "white space" which is one or more of the characters: space, comma, exclamation point. There are three types of data values: integer, real, and character. Each type has a required format which is described below. In the table definitions that follow, the symbols <i>, <r>, and <c> will denote columns that take integer, real, and character data.
respectively. These editorial symbols are not placed in the input file.

Some tables have default values which are used whenever a column is absent or when a row contains no value in a particular column. Since the relational table syntax uses no punctuation (just white space), a "place holder" symbol must be used whenever a value in a row is missing. The place holder symbols are "$" and "?", meaning "not applicable" and "unknown", respectively.

2.1. Integer data values

Integer data are entered as a signed integer number such as 10 or -3. For very large integers, scientific notation may be used in the following form: <sign><mantissa>E<exponent>. Examples of the scientific form are 1E6 (one million) and -2.3E7 (-23 million). In the scientific form, the given value must be integer after the application of the exponent. If a non-integer value is entered, an error message is produced and the program terminates.

2.2. Real data values

Real data are entered using any of the forms provided for integer data and also forms that have fractional parts, such as 2.3 and -3.487E2.

2.3. Character data values

Character data may be entered several ways. When only a single word is to be entered it may be written without surrounding quote marks provided that the word contains no "white space" characters and does not begin with $ or ?. For example, the character data values RED and BLUE may be entered this way. Character data may also be enclosed either by " or by '. The same quote character must both begin and end the character string. The other quote character may be used within the string. Character string data may be up to 80 characters long though some parameters receive their value from only the first character or the first 10 characters in the string. The following strings are legal.

onestrifthoutwhitespace
"A string containing 'apostrophe' characters"
' A string containing "quote" characters'
3. Relational table formats

3.1. Title table

The "title" table is used to provide the caption for the data analysis experiment. The title lines are written at the top of the output report. A sample title table is shown below. The word entered under "TYPE" is ignored in the present implementation. You may enter as many lines of title data as you want.

<table>
<thead>
<tr>
<th>TITLE</th>
<th>TEXT</th>
<th>{table identifier}</th>
</tr>
</thead>
<tbody>
<tr>
<td>TYPE</td>
<td>&lt;&lt;E&gt;</td>
<td>{column-headings}</td>
</tr>
<tr>
<td>MAIN</td>
<td>&quot;ANALYSIS OF DISEASE DATA*&quot;</td>
<td>{editorial symbols denoting data type}</td>
</tr>
<tr>
<td>SUBTITLE</td>
<td>&quot;CHARCOAL ROT AND ROOT ROT&quot;</td>
<td>{1st data row}</td>
</tr>
</tbody>
</table>

CLUSTER/2 is initialized so that it is prepared to receive rows of the TITLE table without the need to give the table-identifier and the column headings (as if the table identifier and column-headings of the TITLE table had already been read in). This feature allows the input file to begin with the first title data line itself but also allows for the complete entry of the TITLE table exactly as given above. An input file that begins as shown below utilizes the implied definition of the TITLE table.

| MAIN | "THIS IS MY MAIN-TITLE*" |
| SUB | "THIS IS MY SUB-TITLE" |

3.2. Parameters table

The PARAMETERS table indicates what you want CLUSTER/2 to do. This table must precede all other tables except TITLE. Each line of the PARAMETERS table requests a full iteration of the clustering algorithm, specifying possibly different cluster organizations (e.g., hierarchical or flat), numbers of clusters, and clustering criterion values, etc. Two simple PARAMETERS tables are shown below.

They both rely on default settings for almost all parameters.

<table>
<thead>
<tr>
<th>PARAMETERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>K</td>
</tr>
<tr>
<td>&lt;t&gt;</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
</tbody>
</table>

{This table implies a flat (one-level) cluster organization and asks for two such clusterings, one with 3 clusters and one with 4 clusters. The user must define the "ALPHA" clustering quality criterion (see section 3.3).}
PARAMETERS
COVERTYPE CRITERION
<e> <e>
HIERARCHICAL Q23

(This table requests a hierarchical cluster organization. CLUSTER/2 will build a classification hierarchy using the clustering quality criterion "Q23" (which must be further defined). The number of clusters formed at each level of the hierarchy will be in the default range of 2 to 4 clusters and no further partitioning will be done for clusters that contain no more than 4 events.)

The PARAMETERS table has many other columns which are optional. The following table shows all columns that are available along with the default values for each column.

<table>
<thead>
<tr>
<th>PARAMETERS</th>
<th>K</th>
<th>MAXK</th>
<th>MAXCRIT</th>
<th>TRACE</th>
<th>B</th>
<th>HI</th>
<th>E</th>
<th>METHOD</th>
<th>WINDOW</th>
<th>COVERTYPE</th>
<th>BASE</th>
<th>PROBE</th>
<th>MAXHEIGHT</th>
<th>MINSIZE</th>
<th>BETA</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;i&gt; &lt;i&gt; &lt;i&gt; &lt;i&gt; &lt;i&gt; &lt;i&gt; &lt;i&gt;</td>
<td>&lt;i&gt;</td>
<td>&lt;i&gt;</td>
<td>&lt;i&gt;</td>
<td>&lt;i&gt;</td>
<td>&lt;i&gt;</td>
<td>&lt;i&gt;</td>
<td>&lt;i&gt;</td>
<td>&lt;i&gt;</td>
<td>&lt;i&gt;</td>
<td>&lt;i&gt;</td>
<td>&lt;i&gt;</td>
<td>&lt;i&gt;</td>
<td>&lt;i&gt;</td>
<td>&lt;i&gt;</td>
<td></td>
</tr>
<tr>
<td>1 1 4 ALPHAN</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>RANDOM</td>
<td>FAST</td>
<td>DISJOINT</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The editorial symbols denoting type (e.g., <i>, <e>) are not part of the table and are never written in the input. The table can contain as many lines of data as desired. The columns are defined below.

**BASE**

BASE and PROBE control the clustering stopping criterion. For each number of clusters considered in finding a solution, CLUSTER/2 always forms BASE number of clusterings. Then PROBE more clusterings are generated (see PROBE below). If any of the PROBE number of clusterings is better than any previous clustering the better cluster description is remembered and another PROBE number of clusterings are generated. The algorithm works such that for each clustering a unique set of seed events is selected. When PROBE clusterings are generated without producing one that is better than a previously obtained one, the algorithm stops and reports the descriptions of the BASE number of best clusterings that were obtained. Increasing BASE increases the number of alternative clusterings that are reported.

**BETA**

BETA specifies the relative importance of a good fit of the cluster descriptions to the data for different numbers of clusters formed. The system chooses the optimized number of clusters by measuring the fit multiplied by the number of clusters raised to the BETA power. This measure has been denoted S and is defined:

\[ S = \text{sparseness} \times k^B \]

where sparseness is the measure of fit and k is the number of clusters. The optimal clustering is the one with the minimum value of S. Increasing BETA increases improvement of fit demanded for greater numbers of clusters (i.e., it tends to cause CLUSTER/2 to select clusterings with fewer clusters).

**COVERTYPE**

COVERTYPE specifies the desired organization of clusters. The specification DISJOINT causes the generation of a "flat," single-level clustering having a particular number of clusters, or the optimized number selected from a specified range. DISJOINT clusters do not overlap; each input event occurs in at most one cluster. The specification INTERSECTING causes the generation of a "flat," single-level clustering but with clusters that may intersect. This feature is not present in the current implementation. The specification HIERARCHICAL causes the generation of a multi-level hierarchy of clusters. The clustering algorithm is applied recursively to clusters formed until either the hierarchy contains a number of levels equal to parameter MAXHEIGHT or the number of clusters to be partitioned falls to have more than
CRITERION specifies the name of the relational table defining a criterion of clustering quality. Such tables are described in section 3.3.

Parameters H1 to H3 control search heuristics. If these parameters are set to "infinity" (this cannot actually be done) then the results of the search for the best clustering are absolutely optimal. Since only trivial problems can be handled in this manner, given the combinatorial explosion that occurs, H1 to H3 are assigned small values and the heuristic search process yields a good, optimized solution but one that is not necessarily optimal. H1 specifies the number of potential cluster descriptions maintained during the process of generating alternative cluster descriptions. In terms of the clustering algorithm (see [4]), H1 gives the number of complexes kept in each partial star. Increasing H1 causes more descriptions to be considered. Values greater than 10 are usually unwarranted, as they do not generally yield an improved solution.

Parameter H2 specifies the maximum number of potential cluster descriptions produced for each cluster. The actual number maintained may be smaller than H2 since the search tree is tapered on one side and has fewer alternative solutions. In terms of the clustering algorithm, H2 gives the maximum number of complexes in a completed star. The effects of H2 are similar to those of H1. Typical values are also less than 10.

Parameter H3 controls the persistence of the search for optimized clusterings. The method inspects clusterings according to their Path-Rank-Order [4]. When the number of successively ordered clusterings inspected without finding a new, most optimized clustering reaches H3, the search terminates. Increasing H3 extends the search, possibly enabling it to find better results.

INITMETHOD specifies the technique to be used to select the first seed events. In the current implementation, the only value allowed is RANDOM.

K specifies the desired number of clusters. The value given for K is placed into both MINK and MAXK (see below).

MAXK specifies the maximum number of clusters to form when generating a clustering. The algorithm tries clusterings containing a number of clusters ranging from MINK to MAXK.

MINK specifies the minimum number of clusters to form.

MAXHEIGHT specifies the maximum allowed height of a cluster hierarchy.

MINSIZE specifies the minimum size (in terms of number of events) of a cluster that is subject to recursive clustering into smaller parts. Only clusters that contain more than MINSIZE number of events are further partitioned.

NIDSPEED specifies the technique used to make Non-disjoint clusters Into Disjoint ones. The value FAST causes CLUSTER/2 to use seed events as well as previously determined cluster descriptions to form an "against set" for building alternative cluster descriptions for clusters. This improves the speed of the program but places high weight on the clusters that happen to be formed first. This may lead to less-optimized results. The value SLOW causes CLUSTER/2 to use a cluster-adjusting algorithm (NID) whenever the clusters created using only seed events as the "against set" intersect one another. The cluster-adjustment algorithm adds to the solution
time but the clusters are not weighted according to the order in which they are formed and can be more optimized.

**PROBE**

PROBE controls the persistence of the cluster formation process. The system forms a candidate clustering and compares it (according to the clustering quality criterion) to previously generated clusterings. The system performs PROBE number of such cycles looking for a better clustering. Each time such a clustering is found, another PROBE cycles are performed until PROBE number of successive cycles fail to find an improved solution. The algorithm also halts if all combinations of events have been considered as seeds. Increasing PROBE permits the program to explore more combinations of seed events and may permit the program to discover better clusterings.

**TRACE**

The TRACE column controls the generation of intermediate clustering descriptions in the output report file. The value OFF limits output just to the final cluster descriptions. The value ON requests the output of supplemental descriptions for each intermediate clustering that is produced.

**PRINT**

The PRINT column may be defined in lieu of the TRACE column. The values entered under the PRINT column explicitly control the printout contents. The value given is a single character string composed of the letters A, C, E, I, N, P, T, V. Except for letter T, the order of the letters is unimportant. The letter I controls the printing of intermediate results and is totally equivalent to the ON setting of the TRACE option (above). If letter I is not present in a particular row of the PARAMETERS table, that part of the run is made with no printing of intermediate results. Each letter activates a different type of printout during the running of CLUSTER/2. If the PARAMETERS table contains many rows of data, the system processes the values in the PRINT column as a single specification, as if they were concatenated to form a single string of letters. Each letter has the following meaning:

- **A** - enable all print functions
- **C** - print the input -CRITERION tables
- **E** - print the input EVENTS table
- **I** - print intermediate results
- **N** - print the input -NAMES, -ONAMES, -STRUCTURE tables
- **P** - print the PARAMETERS table
- **T** - print the TITLE table ONLY; this negates all other options
- **V** - print the DOMAINS and VARIABLES tables

The initial setting is PC which normally prints PARAMETERS and -CRITERION tables. Giving a value such as E (or CFE) adds to the current print option the request to print the EVENTS table. All options except T are additive; T cancels all optional printout except for the TITLE data. The value TE first eliminates the current (PC default) option and then establishes just the E option, causing only the EVENTS table to be printed with the TITLE data that is always printed.

The PARAMETERS table directs the work performed by CLUSTER/2. Any combination of parameter values that is to be performed is entered as a separate row. The clustering algorithm is repeated for each row, using the parameters given on that row.
3.3. Criterion table

The -CRITERION table form may appear more than once in the input file. Each occurrence is given a unique name which consists of a specific part followed by a general part (-CRITERION). There must be a -CRITERION table for each name specified in the CRITERION column of the PARAMETERS table. From the previous PARAMETERS examples the table ALPHA-CRITERION and possibly the table Q23-CRITERION would need to be entered, depending on which names (ALPHA or Q23 or others) were used in the PARAMETERS table. Either one of the following two tables could be used to define the "ALPHA" criterion.

<table>
<thead>
<tr>
<th>CRIT#</th>
<th>TOL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.3</td>
</tr>
<tr>
<td>2</td>
<td>0.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CRIT#</th>
<th>CRITERION</th>
<th>TOL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PSPARENESS</td>
<td>0.3</td>
</tr>
<tr>
<td>2</td>
<td>DISJOINTNESS</td>
<td>0.0</td>
</tr>
</tbody>
</table>

The program applies the elementary evaluation criteria in a user-defined order. The column # specifies the order of application of the elementary criteria specified in the CRIT column. The numbers in the # column must be in numerical order, starting with 1.

Either the CRIT# column or the CRITERION column is used to specify which elementary criterion is applied. If the CRIT# column is used, the criteria must be indicated by criterion number. If the CRITERION column is used, the criteria must be indicated by criterion name. The elementary criterion numbers may be taken from the following definitions (see sec 3.3.1). The elementary criteria are combined according to the values specified in the -CRITERION table to form the Lexicographical Evaluation Functional (LEF) [5] used to evaluate the clustering quality.

The CRITERION column may be used in lieu of the CRIT# column. When using CRITERION, the elementary criteria are identified by name, rather than by number (see sec 3.3.1).

TOL is a positive real number, normally less than or equal to 1.0. If TOL is larger than 1.0, it is an absolute tolerance. When evaluating the LEF, clusterings within this amount of the best one are judged to be equivalent. If TOL is less than or equal to 1.0, the tolerance is taken as the indicated fraction of the difference between the best and worst clustering scores for the elementary criterion. For example, with TOL set to 0.3 then clusterings scoring within 30% of the best clustering (on a scale from the best score to the worst score) are judged to be equivalent. A TOL of 1.0 causes all clusterings to be judged the same; a TOL of 0.0 causes no clusterings to be judged the same unless their scores are absolutely identical.
TOLERANCE is a synonym for TOL.

3.3.1. Elementary criteria available in CLUSTER/2

The following criteria are defined in CLUSTER/2. These criteria are component parts of a user-specified Lexicographical Functional with tolerances (LEF) [5]. They are specified in a -CRITERION table by their index number or name. CLUSTER/2 optimizes the generated clusterings by favoring clusters whose descriptions have the lowest scores for the elementary criteria specified in the LEF. If a negative elementary criteria index is given, or if "-" is placed directly in front of the name, the negative of the score is used. As CLUSTER/2 favors clusters with descriptions that have the lowest negatives, the effect is the same as favoring clusters whose descriptions have the highest scores.

The elementary criteria are presented in order by criterion index number. The "input specification" annotation gives the keyword used in the CRITERION column. Only the unique part of the name (capitalized) need be entered.

1. Sparseness (input specification: SParseness)

The sparseness of clustering is the sum of the sparsenesses of the complexes which comprise the clustering. The sparseness of a complex is the number of events it covers which are not given in the input data. Such events are called "unobserved" events. The sparseness is computed by calculating the "area" of the complex and subtracting from it the number of points which correspond to given (observed) events. The "area" of a complex is the number of points (each corresponding to a possible event) in the subset of the event space covered by the complex.

Example: Suppose that complex \([x_1=2,4][x_3=4,7]\) covers 4 observed events. (assume the variables are \(x_1, x_2, x_3\), and that the domain of \(x_2\) contains 3 values)

The "area" of the complex is the product of the numbers of values in the references for each variable. In this example, area = 2 x 3 x 4 = 24 (2 values for \(x_1\), 3 values for \(x_3\) — its entire domain, 4 values for \(x_3\)). The sparseness of this complex is "area" - #observed events or 24 - 4 = 20.

2. Disjointness (input specification: DI斯jointness)
The disjointness (degree of intersection) of a clustering is the sum of the degrees of intersection of each pair of complexes in the clustering. The degree of intersection of two complexes is the total number of selectors that involve the same variable and have intersecting reference sets. The following example involves three variables $x_1, x_2, x_3$.

Example:

$\alpha_1: [x_1=1.3][x_2=4]$  
$\alpha_2: [x_1=2.5][x_2=2]$  
$\alpha_3: [x_1=4][x_2=3][x_3=2]$

The degree of intersection of complex $\alpha_1$ with $\alpha_2$ is 1 because the references used with variable $x_1$ in each complex contain intersecting values and the selectors omitted for $x_3$ also intersect (a "dropped" selector is equivalent to one with all possible domain values in the reference list). The degree of intersection of complex $\alpha_3$ with $\alpha_1$ is 2 because two selectors (for $x_3$) intersect. (In $\alpha_1$ the missing selector for variable $x_3$ is presumed to have a reference set equal to the entire domain of $x_3$).

3. Number of events occurring in more than 1 complex (input specification: Multicov)

This criterion is for use only when considering intersecting clusters. It is not used in the current implementation.

4. Balance (input specification: Balance)

This criterion measures the unevenness in the cluster populations. It is the sum of the deviations from uniform distribution of observed events in each complex. For one complex, this deviation is calculated as the absolute value of the difference between the actual number of observed events covered and $1/k$ of the total number of the observed events, where $k$ is the number of complexes (clusters).

5. Commonality (input specification: Commonality)

This criterion measures the number of common attributes in the cluster descriptions by counting the number of selectors in all clusters (dropped selectors are not counted). The negative of the number of selectors is used in order to maximize the commonality by minimizing the numerical score.
6. Dimensionality (input specification: DIMensionality)

This criterion measures the number of variables which take on different values in every complex. The number reported for each complex is the number of variables in the complex which occur in selectors in all complexes of the clustering with mutually disjoint reference values.

7. Simplicity (input specification: Simplicity)

The simplicity of a clustering is the sum of the simplicities of all selectors contained in all complexes in the clustering. The simplicity of one selector is the sum of the cost of the variable plus the cost of the reference list. Variable costs are provided by the user via the VARIABLES table (see Sec. 3.4) or given the default cost of 1. Reference list costs are computed according to the type of the domain. For structured domains, the cost is 0. For linear domains, the cost is 0 if only one value occurs in the reference list, otherwise the cost is 1. For nominal domains, the cost is the number of values in the reference list minus 1.

8. Projected Sparseness (input specification: Pprojected sparseness)

Projected sparseness is calculated just like regular sparseness, but only some of the dimensions of the event space are considered. When calculating the "area" used in calculating projected sparseness, only the variables which are present (not dropped) in at least one complex of the clustering are considered.

This is equivalent to imagining that the domain sizes of all universally-dropped variables is just 1.

Example: Suppose that \(|x_1=2,4||x_3=4..7|\) covers 4 observed events.

The factors used in computing the "area" of the complex depend, in part, on other complexes in the clustering. If, for this example, the variable \(x_5\) has a domain containing 6 values and appears in a selector in some cluster description then it is used to calculate the "area" of the complex as follows:

\[
\text{"area"} = 2 \times 4 \times 6
\]

where 2 and 4 are the numbers of values in the reference lists for variables \(x_1\) and \(x_3\), respectively, and 6 is the number of levels in variable \(x_5\).
The projected sparseness is then \((2 \times 4 \times 6) \cdot 4 = 48\). Projected sparseness may be negative.

9. Number of exceptional events (input specification: Except)

Certain situations cause the formation of a clustering which fails to cover all the events. Any events not covered are placed into an exceptions category, which is shown on the printout. Most elementary criteria are applied to each cluster description (i.e., to each complex) and the sum over all cluster descriptions is used to evaluate an entire clustering. The number of exceptional events criterion is an exception and is applied only to the entire clustering. When applied to individual complexes, this criterion produces a score of 0 since the exceptional events list is a property of a clustering, not an individual cluster.

3.4. Domains table

The DOMAINS table is used to define variable domains that are likely to apply to several variables. Variables may be defined entirely with a VARIABLES table (described next) or with the help of the DOMAINS table. When many variables have exactly the same domain type and number of values, it is more convenient to define the domain first, and then reference it in the VARIABLES table. A simple domain definition table is shown below.

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>LEVELS</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;e&gt;</td>
<td>&lt;e&gt;</td>
<td>&lt;4&gt;</td>
</tr>
<tr>
<td>COLOR</td>
<td>NOM</td>
<td>4</td>
</tr>
<tr>
<td>SIZE</td>
<td>LIN</td>
<td>3</td>
</tr>
</tbody>
</table>

NAME specifies the name of the domain. In the sample table above, two different domains are defined: color and size.

TYPE specifies the type of the domain. Only the first letter of the word given is significant: N means nominal, L means linear, S means structured. See the discussion of TYPE below for further information.

LEVELS gives the number of distinct values in the domain. If the number of levels is \(j\), the domain consists of the integer values from 0 to \(j-1\).

An optional column COST may be used to indicate the cost of a variable having this domain.
3.5. Variables table

The VARIABLES table is used to describe the attribute variables used to describe each event. A simple variables table is shown below.

<table>
<thead>
<tr>
<th>#</th>
<th>TYPE</th>
<th>LEVELS</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;1&gt;</td>
<td>&lt;c&gt;</td>
<td>&lt;t&gt;</td>
</tr>
<tr>
<td>1</td>
<td>NOM</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>LIN</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>LIN</td>
<td>5</td>
</tr>
</tbody>
</table>

The # column gives the variable ordinal. The above table defines the variables X₁, X₂, and X₃. The ordinal values must be in ascending order, starting with 1.

TYPE specifies the type of the domain of each variable. Only the first letter of the word given under TYPE is significant. Nominal (or just N) denotes an unordered qualitative domain. Linear (or just L) denotes an ordered quantitative domain. Structured (or just S) denotes a tree-ordered (or graph-ordered) domain. If the Structured domain type is specified, a -STRUCTURE table must be entered, see below.

LEVELS gives the number of distinct values in the domain. If the number of levels is j, CLUSTER/2 requires that the data values be integers from 0 up to j-1. The event descriptions are entered as a vector of integers, one integer for each variable.

In printing the descriptions for each cluster, CLUSTER/2 refers to the variables as X₁, X₂, X₃, etc. The name of the variable in the report can be changed by adding a NAME column to the VARIABLES table. The following table defines the variables SHAPE, SIZE, WEIGHT.

<table>
<thead>
<tr>
<th>#</th>
<th>TYPE</th>
<th>LEVELS</th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;1&gt;</td>
<td>&lt;c&gt;</td>
<td>&lt;t&gt;</td>
<td>&lt;t&gt;</td>
</tr>
<tr>
<td>1</td>
<td>NOM</td>
<td>4</td>
<td>SHAPE</td>
</tr>
<tr>
<td>2</td>
<td>LIN</td>
<td>3</td>
<td>SIZE</td>
</tr>
<tr>
<td>3</td>
<td>LIN</td>
<td>5</td>
<td>WEIGHT</td>
</tr>
</tbody>
</table>

Another optional column is COST which can accept any real value. COST is used in calculating the score for elementary criterion number 7, the "simplicity" of a cluster description. If the COST column is not entered, all variables have the default cost of 1.

To indicate that a variable has a domain identical to a predefined domain (defined via the DOMAINS table) the name of the variable is entered in a special way (this requires that the NAME column be present in the VARIABLES table). The name is entered as <variablename>-<domainname>, i.e., as one character string composed of the desired variable name, a hyphen, and a
domain name that was defined in the DOMAINS table. When this is done, the TYPE, LEVELS, and
COST for the variable are taken from the domain definition regardless of whether specified in the
VARIABLES table. If any defined domain has associated -NAMES, -ONAMES, or -STRUCTURE tables
(all described below) then these tables must precede the VARIABLES table.

3.6. Events table

The EVENTS table holds the attribute vectors for all objects or situations being studied. Each
object or situation is defined by one row of integer values (subsequent sections tell how to handle symbolic
attribute values). Each attribute value must be placed in its corresponding column and must be in the
range from 0 up to LEVELS-1, where LEVELS denotes the number of levels for that attribute as specified
in the VARIABLES table. The headings for the EVENTS table are the names of the variables. If the
variables are defined without the NAME column option, the variable names are of the form Xn, where n
is the variable ordinal (e.g., X1). If the variable is defined with the NAME column, the given name is
used without any hyphen or domain name part (e.g., SHAPE). The following EVENTS table shows the
form used with variables that are not given a NAME.

EVENTS
-13-

<table>
<thead>
<tr>
<th>#</th>
<th>X1</th>
<th>X2</th>
<th>X3</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;1&gt;</td>
<td>&lt;1&gt;</td>
<td>&lt;1&gt;</td>
<td>&lt;1&gt;</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

The same table must be entered in the following form if the variables are given names. The heading
keywords must match the values from the NAME column of the VARIABLES table (they need not be in
the same order).

EVENTS

<table>
<thead>
<tr>
<th>#</th>
<th>SHAPE</th>
<th>SIZE</th>
<th>WEIGHT</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;1&gt;</td>
<td>&lt;1&gt;</td>
<td>&lt;1&gt;</td>
<td>&lt;1&gt;</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

The # column gives the event number. The values in this column must be in sequence and must begin
with the value 1.
When working with events described by very long attribute vectors, the number of columns in the event table (one column per attribute) may make the tables unmanageably wide. CLUSTER/2 permits such tables to be split into left and right halves (or left, middle, right, etc.). To make use of this feature, the EVENTS table is specified several times, with different column-headings (for different variables) used in each EVENTS table. (Each table contains the # columns). All of the separate EVENTS tables must have the same number of rows and the rows must be in exactly the same order. CLUSTER/2 collects the values from the ith row of each table into the attribute description of the ith event.

The optional column WT may be entered to provide a weight for each event. The weight may be used to indicate the relative frequency of observation of the event. The use of weighted events alters the calculation of sparseness. Each observed event counts according to its weight, so a complex with an "area" of 4 and covering 2 observed events receives a sparseness of 4 minus the combined weights of the covered events. With large weights, the sparseness calculation may produce negative values (this causes no problems). Negative sparseness should be interpreted as a measure of fit something like weight density. The more weight of covered events, the more negative the sparseness.

3.7. NAMES table

A -NAMES table may be entered for any variable or domain definition. The specific part of the table name is the name of the variable (either Xn form or as given in the NAMES column of the VARIABLES table) or domain. The -NAMES table specifies a symbolic name to be used in lieu of the integer variable value in both the reported cluster descriptions and the input of the event data (i.e., values for this variable in an EVENTS table must be symbolic rather than integer). The -NAMES table is illustrated below.

```
SHAPE-NAMES
VALUE  NAME
<1>    <c>
0       SQUARE
1       RECTANGLE
2       CIRCLE
3       OVAL
```

For linearly ordered variables and domains, the VALUE integer establishes the ordering of symbols within the domain. If a single value denotes a range of observed features (e.g., the value 3 denotes a temperature
in the range from 68 to 72) the name should be written "lowname..highname" (e.g., "68..72"). The use of
the .. range indicator allows CLUSTER/2 to combine names appropriately when reporting an interval
value of a linearly ordered variable.

3.8. ONAMES table

The -ONAMES table may be entered in lieu of a -NAMES table. The -ONAMES table specifies
output-only names for the integer values of the domain of the variable. If the input data is numerical, an
-ONAMES table will allow the output to be symbolic while the input remains numerical. (Using the
-NAMES table requires the input to be symbolic rather than numerical.) An -ONAMES table contains the
same columns as a -NAMES table.

3.9. Structure table

Variables that are defined as Structured TYPE have a tree-structured (or graph-structured)
domain. The LEVELS value given in the VARIABLES table (or DOMAINS table) refers to the number
of discrete leaf values in the tree (or graph). Higher-order node values (generalized values of the variable)
are defined by the -STRUCTURE table. The specific part of the table name is the name of the variable
(either the Xn form or as given in the NAME column of the VARIABLES table).

The -STRUCTURE table takes two forms, depending on the use of an associated -NAMES table.
If no -NAMES table is used for the variable, the -STRUCTURE table looks like this:

| SHAPE-STRUCTURE | VALUE | SUBVALUE | (SUBVALUE) | (SUBVALUE) | ...
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>4</td>
<td>2</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

The VALUE columns and the first SUBVALUE column are required. Additional SUBVALUE columns
may be used by including SUBVALUE several times in the column-heading list. (In the above illustration
the parenthesis around SUBVALUE denotes that the column is optional. The parenthesis is not used in
the input file.) For each row in the -STRUCTURE table, the value in the VALUE column is defined as
the parent of the values in the SUBVALUE columns. Thus each row of the -STRUCTURE table is
defining one (or more) links in the tree-structured domain. Any nodes or leaf values that have no
explicitly defined parent are implicitly parented by the root of the tree. The values that occur under
VALUE must be greater than any regular (leaf) value for the variable. If the LEVELS column of the VARIABLES table specifies j levels, the leaf values in the domain are denoted by values from 0 to j-1, and the values j, j+1, j+2, ... could appear in the VALUE column of the -STRUCTURE table to denote generalized domain values. In the example above, the variable SHAPE has four levels (values 0 to 3).

Two new values (4 and 5) are defined as generalizations of values 2 and 3 and values 0 and 1, respectively.

If a -NAMES table is given for the variable (the -NAMES table must precede the -STRUCTURE table for the same variable) the following form of the -STRUCTURE table is to be used:

```
SHAPE - STRUCTURE
NAME     SUBNAME     SUBNAME
<<        <<         >>
4-SIDED   SQUARE      RECTANGLE
CIRCULAR  CIRCLE      OVAL
```

The tree on the right illustrates the semantics of this -STRUCTURE table. This latter form of the -STRUCTURE table has all the options of the previous form, but with symbolic (character data) names for all values. The names given to the defined generalized domain values must be unique among all names (including leaf names) in the same domain.

If a generalized value must be the parent to a great number of subvalues (or subnames) the table may become unmanageable. In such cases it is permitted to repeat the VALUE (or NAME) entry on adjacent lines, giving part of the subvalue (subname) list on each line. CLUSTER/2 merges the definitions together into one specification of a generalized value for all of the subvalues mentioned.

4. A sample input file

The following sample input file makes use of the relational table forms described above. Some comments about the interpretation of this example are given below.

```
MAINTITLE "MICROCOMPUTERS"

PARAMETERS
  K  TRACE  CRITERION NIDSPEED COVERTYPE
  2  ON     DS      SLOW   DISJOINT
  $ OFF    FC      FAST   HIERARCHICAL
```
### DS-CRITERION

<table>
<thead>
<tr>
<th>CRITERION</th>
<th>TOLERANCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIS</td>
<td>0.3</td>
</tr>
<tr>
<td>COM</td>
<td>0.0</td>
</tr>
<tr>
<td>PSPARS</td>
<td>0.0</td>
</tr>
</tbody>
</table>

### FC-CRITERION

<table>
<thead>
<tr>
<th>CRITERION</th>
<th>TOLERANCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSPARS</td>
<td>0.3</td>
</tr>
<tr>
<td>SIM</td>
<td>0.0</td>
</tr>
</tbody>
</table>

### VARIABLES

<table>
<thead>
<tr>
<th>#</th>
<th>NAME</th>
<th>TYPE</th>
<th>LEVELS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>MP</td>
<td>STR</td>
<td>13</td>
</tr>
<tr>
<td>2</td>
<td>RAM</td>
<td>LIN</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>ROM</td>
<td>LIN</td>
<td>7</td>
</tr>
<tr>
<td>4</td>
<td>DISP</td>
<td>STR</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>KEYS</td>
<td>LIN</td>
<td>5</td>
</tr>
</tbody>
</table>

### MP-NAMES

<table>
<thead>
<tr>
<th>VALUE</th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>8080A</td>
</tr>
<tr>
<td>1</td>
<td>6502</td>
</tr>
<tr>
<td>2</td>
<td>Z80</td>
</tr>
<tr>
<td>3</td>
<td>1802</td>
</tr>
<tr>
<td>4</td>
<td>6502C</td>
</tr>
<tr>
<td>5</td>
<td>6502A</td>
</tr>
<tr>
<td>6</td>
<td>68000</td>
</tr>
<tr>
<td>7</td>
<td>6800</td>
</tr>
<tr>
<td>8</td>
<td>6805</td>
</tr>
<tr>
<td>9</td>
<td>6809</td>
</tr>
<tr>
<td>10</td>
<td>8048</td>
</tr>
<tr>
<td>11</td>
<td>Z8000</td>
</tr>
<tr>
<td>12</td>
<td>HP</td>
</tr>
</tbody>
</table>

### MP-STRUCTURE

<table>
<thead>
<tr>
<th>NAME</th>
<th>SUBNAME</th>
<th>SUBNAME</th>
<th>SUBNAME</th>
<th>SUBNAME</th>
<th>SUBNAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>6502X</td>
<td>6502</td>
<td>6502C</td>
<td>6502A</td>
<td>$</td>
<td></td>
</tr>
<tr>
<td>8080X</td>
<td>8080A</td>
<td>Z80</td>
<td>8048</td>
<td>$</td>
<td></td>
</tr>
<tr>
<td>8BIT</td>
<td>6502X</td>
<td>8080X</td>
<td>1802</td>
<td>6800</td>
<td></td>
</tr>
<tr>
<td>8BIT</td>
<td>6805</td>
<td>6809</td>
<td>$</td>
<td>$</td>
<td></td>
</tr>
<tr>
<td>16BIT</td>
<td>68000</td>
<td>Z8000</td>
<td>HP</td>
<td>$</td>
<td></td>
</tr>
</tbody>
</table>

### RAM-NAMES

<table>
<thead>
<tr>
<th>VALUE</th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>16K</td>
</tr>
<tr>
<td>1</td>
<td>32K</td>
</tr>
<tr>
<td>2</td>
<td>48K</td>
</tr>
<tr>
<td>3</td>
<td>64K</td>
</tr>
</tbody>
</table>
ROM-ONAMES
VALUE NAME
0 1K
1 4K
2 8K
3 10K
4 11K..16K
5 26K
6 80K

DISP-ONAMES
VALUE NAME
0 TERMINAL
1 "B/W_TV"
2 "COLOR_TV"
3 BUILT-IN

DISP-STRUCTURE
NAME VALUE SUBVALUE SUBVALUE
TV 4 1 2

KEYS-ONAMES
VALUE NAME
0 52
1 53..56
2 57..63
3 64..73
4 92

EVENTS
# MP RAM ROM DISP KEYS
1 6502 2 3 2 0
2 6502 2 3 2 2
3 6502A 1 4 2 3
4 Z80 2 1 1 2
5 8080A 3 0 3 3
6 Z80 3 2 3 3
7 HP 1 6 3 4
8 Z80 3 2 0 2
9 6502 1 3 1 1
10 6502C 2 3 1 1
11 Z80 2 4 1 1
12 Z80 2 4 3 3

<end of file>

The tables used in the above file are PARAMETERS, DS-CRITERION, FC-CRITERION, VARIABLES, MP- NAMES, MP-STRUCTURE, RAM- NAMES, ROM- NAMES, DISP- NAMES, DISP- STRUCTURE, KEYS- NAMES, and EVENTS. The entire clustering algorithm is to be performed two times (because there are two rows in the PARAMETERS table). The two different user-defined criterion names (DS and
FC) have -CRITERION tables which define the elementary criteria used. The variables MP, RAM, ROM, DISP, and KEYS have names associated with the integer data values. There is a -NAMES table for each variable. The variables MP and DISP have structured domains. There is a -STRUCTURE table for each of these two variables.

The table MP-STRUCTURE presents a graph-structured domain in which the generalized values 6502x (members of the 6502 family of microprocessors), 8080x, 8bit, and 16bit are defined. The inset shows the graph-structured domain of the DISP variable as defined by the DISP-STRUCTURE table.

5. Program output corresponding to sample input file

The beginning of the output listing shows the PARAMETERS table and the two -CRITERION tables. Since the PARAMETERS table contains two data rows, the CLUSTER/2 program will be invoked twice, once for k=2 for a "flat" clustering, and once for a hierarchical clustering in which the best k in the range 2 to 4 will be determined (using the default beta value of 3.0). Parts of the remaining listing that are of particular interest have been marked with circled numbers on the right-hand side.

Location "1" in the listing marks the beginning of the first clustering, which is identified as "experiment 1." The criterion applied is "ds" as defined by the DS-CRITERION table. The number of clusters is set (by the user) to 2. Because the trace mode is on (see first data row of the PARAMETERS table), each clustering that is formed is described in the listing under the heading "intermediate results...". The column "iter" gives the clustering iteration number while the column "cplx#" gives the number of each cluster formed (here, only two clusters will be formed). Under the column "vl-rule" is the description of the cluster using the names specified in the various -NAMES tables. In iteration 1, the first cluster consists of microcomputer systems that use an 8bit microprocessor (mp), that have 32K to 48K of ram, that have 10K to 16K of rom, that use a TV for a display device, and that have 52 to 50 keys on their keyboards. The column "events covered" indicates that this first cluster covers input events (rows in the EVENTS table) numbered 1, 0, 10, and 11. The "costs" columns show the scores obtained by each cluster on the elementary criteria in the criterion of clustering quality defined by the -CRITERION table for the "ds" criterion. A "totals" row on the listing shows the total scores for the entire clustering.
When clustering quality improvement ceases, CLUSTER/2 reports the best clusterings obtained. For the "experiment 1" this report is at location "2" in the listing. Because the default value for BASE is used (see PARAMETERS table), two alternative clusterings are shown, with the best one listed first. Since the first experiment specified k=2 and a disjoint (flat) cover type, the processing for experiment 1 is complete.

Location "3" marks the beginning of "experiment 2" which uses a different clustering quality criterion (called "fc" by the user). In experiment 2, a hierarchical clustering will be generated, with k determined by the program upon considering clusterings with k ranging from 2 to 4. The trace option is off for experiment 2, so only the best clusterings appear on the listing. (See the second line of the PARAMETERS table for these parameter settings).

The microcomputers are first clustered into two categories. Then, at locations "4" and "5" in the listing, they are clustered into three and four categories, respectively. At location "6" in the listing, the best value of k is determined by comparing the "S-scores" for each of the clusterings for k=2 to k=4. The S-score is the sparseness multiplied by k raised to the beta power. In this example, beta has the default value of 3.0. For experiment 2, the best top-level clustering in the cluster hierarchy is the one obtained with k=3. Its description is shown at location "6".

Of the three clusters formed at the top level of the hierarchy, one contains just a single event and undergoes no further partitioning. The other clusters contain 5 and 6 events, respectively. The events in the first cluster of the top level of the hierarchy are further clustered, leading to the final result shown at location "7". The best value for k for this second-level clustering was 3. The clusters are too small (number of events not above parameter minsize) to be partitioned any further.

Following location "7", the output is shown for the clustering of the second first-level cluster. The final results for this second-level clustering is shown at location "8" in the listing. With beta set to 3.0, the best clustering is with k=4.
output from conjunctive conceptual clustering program cluster/2  last upgrade: 11/10/83

**parameters**

<table>
<thead>
<tr>
<th>mixmask</th>
<th>trace</th>
<th>h1</th>
<th>h2</th>
<th>h3</th>
<th>initmethod</th>
<th>sidedisp</th>
<th>covertype</th>
<th>criterion</th>
<th>base</th>
<th>probe</th>
<th>beta</th>
<th>maxheight</th>
<th>minsize</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>4</td>
<td>off</td>
<td>3</td>
<td>2</td>
<td>random</td>
<td>fast</td>
<td>disjoint</td>
<td>ds</td>
<td>2</td>
<td>2</td>
<td>3.0</td>
<td>99</td>
<td>4</td>
</tr>
</tbody>
</table>

**ds-criterion**

<table>
<thead>
<tr>
<th>criterion</th>
<th>tolerance</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.30</td>
</tr>
<tr>
<td>2</td>
<td>0.60</td>
</tr>
<tr>
<td>3</td>
<td>0.90</td>
</tr>
</tbody>
</table>

**fc-criterion**

<table>
<thead>
<tr>
<th>criterion</th>
<th>tolerance</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>except</td>
</tr>
<tr>
<td>2</td>
<td>paper</td>
</tr>
<tr>
<td>3</td>
<td>sim</td>
</tr>
</tbody>
</table>

**variables**

<table>
<thead>
<tr>
<th>type</th>
<th>levels</th>
<th>cost</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>structured</td>
<td>18</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>linear</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>linear</td>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>structured</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>linear</td>
<td>5</td>
<td>1</td>
</tr>
</tbody>
</table>

**mp-names**

<table>
<thead>
<tr>
<th>value</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>6000a</td>
</tr>
<tr>
<td>1</td>
<td>6001</td>
</tr>
<tr>
<td>2</td>
<td>2B0</td>
</tr>
<tr>
<td>3</td>
<td>1B02</td>
</tr>
<tr>
<td>4</td>
<td>6000c</td>
</tr>
<tr>
<td>5</td>
<td>6002a</td>
</tr>
<tr>
<td>6</td>
<td>6000b</td>
</tr>
<tr>
<td>7</td>
<td>6006</td>
</tr>
<tr>
<td>8</td>
<td>6005</td>
</tr>
<tr>
<td>9</td>
<td>6009</td>
</tr>
<tr>
<td>10</td>
<td>6008</td>
</tr>
<tr>
<td>11</td>
<td>6000c</td>
</tr>
<tr>
<td>12</td>
<td>600</td>
</tr>
</tbody>
</table>

**mp-structures**

<table>
<thead>
<tr>
<th>value</th>
<th>name</th>
<th>subname</th>
<th>subname</th>
<th>subname</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>6002x</td>
<td>6002c</td>
<td>6002a</td>
<td>6002a</td>
</tr>
<tr>
<td>14</td>
<td>6030x</td>
<td>6030c</td>
<td>6030c</td>
<td>6030c</td>
</tr>
<tr>
<td>15</td>
<td>6011</td>
<td>6012a</td>
<td>6012c</td>
<td>6012a</td>
</tr>
<tr>
<td>16</td>
<td>6011</td>
<td>6013</td>
<td>6012c</td>
<td>6012a</td>
</tr>
</tbody>
</table>

**ran-names**

<table>
<thead>
<tr>
<th>value</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>10k</td>
</tr>
<tr>
<td>1</td>
<td>32k</td>
</tr>
<tr>
<td>2</td>
<td>48k</td>
</tr>
<tr>
<td>3</td>
<td>64k</td>
</tr>
</tbody>
</table>

**rom-names**

<table>
<thead>
<tr>
<th>value</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1x</td>
</tr>
<tr>
<td>1</td>
<td>4x</td>
</tr>
<tr>
<td>2</td>
<td>8x</td>
</tr>
<tr>
<td>3</td>
<td>10k</td>
</tr>
<tr>
<td>4</td>
<td>16k</td>
</tr>
<tr>
<td>5</td>
<td>256x</td>
</tr>
<tr>
<td>6</td>
<td>86x</td>
</tr>
</tbody>
</table>

**disp-names**

<table>
<thead>
<tr>
<th>value</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>terminal</td>
</tr>
<tr>
<td>1</td>
<td>b/u tv</td>
</tr>
<tr>
<td>2</td>
<td>col/c tv</td>
</tr>
<tr>
<td>3</td>
<td>built-in</td>
</tr>
</tbody>
</table>
- 22 -

display structure
value name          submenu          submenu
4        tv
	b/w_tr

text

keys-COLOR
value name
0        53
1        53 65
2        57 63
3        64 73
4        92

events
#   sp    ram    rom    disp    keys
1     5507 48k 10k    color_tv  62
2     6502 48k 10k    color_tv  67 63
3     6502a 32k 10k..10k color_tv  54 73
4     6502 48k 4k    b/w_tr  54 63
5     6502a 64k 4k    built-in  64 73
6     6502 64k 8k    built-in  64 73
7     6502 64k 60k    built-in  20 64 73
8     6502 64k 8k    terminal  57 63
9     6502 32k 10k    b/w_tr  53 64
10    6502 48k 10k    built-in  53 64
11    6502 48k 10k..10k  b/w_tr  53 64
12    6502 48k 10k..10k  built-in  54 73

microprocessors

experience 1: k=2, criterion=de

intermediate results ...

iter/npix   v-rule
1         1        [np>6520] [ram=52k..48k][rom=10k..16k] [disp=color_tv][keys=63..66]
2         2        [np>6520] [ram=32k..64k][keys=57..62]
1        totals (769 ms)

2         1        [np>6520] [ram=32k..48k][rom=10k..16k] [disp=terminal] 0 4
3         2        [np>6520] [ram=48k..64k][rom=1k..16k] 0 0 1604 4 2
1        totals (2666 ms)

3         1        [np>6520] [ram=32k..48k][rom=10k..16k] [disp=terminal] 6 0 1604 4 2
3         2        [np>6520] [ram=48k..64k][rom=1k..16k] [disp=color_tv][keys=53..70]
3       totals (2500 ms)

4         1        [np>6520] [ram=32k..48k][rom=10k..16k] [disp=terminal][keys=92]
4         2        [np>6520] [ram=48k..64k][rom=1k..16k] [disp=color_tv][keys=62..70]
4        totals (2527 ms)

5         1        [np>6520] [ram=32k..48k][rom=10k..16k] [disp=terminal][keys=92]
5         2        [np>6520] [ram=48k..64k][rom=1k..16k][disp=color_tv][keys=62..70]
5        totals (2566 ms)

6         1        [np>6520] [ram=32k..48k][rom=10k..16k] [disp=terminal][keys=92]
6         2        [np>6520] [ram=48k..64k][rom=1k..16k][disp=color_tv][keys=62..70]
6        totals (2567 ms)

the 2 test clusterings follow... (13117 ms)

iter/npix   v-rule
4         1        [np>6520] [ram=32k..48k][rom=10k..16k] [disp=terminal][keys=92]
4         2        [np>6520] [ram=48k..64k][rom=1k..16k][disp=terminal][keys=62..70]
4        totals (2298 ms)

2        1        [np>6520] [ram=32k..48k][rom=10k..16k] [disp=terminal]
2         2        [np>6520] [ram=48k..64k][rom=1k..16k] [disp=terminal][keys=62..70]
2        totals (2298 ms)

(22 states built)

for the best solution above, s = 1.0e-04
### Experiments with k=2, criterion=fc

The 2 best clusterings follow... (13263 ms)

<table>
<thead>
<tr>
<th>Iter/Cluster</th>
<th>Seed</th>
<th>Except</th>
<th>Papers</th>
<th>Sim</th>
<th>Events Covered</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 1</td>
<td>283</td>
<td>0</td>
<td>1184</td>
<td>1</td>
<td>8 1.2,3,7.9,10</td>
</tr>
<tr>
<td>2 2</td>
<td>302</td>
<td>0</td>
<td>294</td>
<td>12</td>
<td>8 4.5,6.8,11.12</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>4 1</td>
<td>297</td>
<td>0</td>
<td>1458</td>
<td>4</td>
<td>7 3.7.9</td>
</tr>
<tr>
<td>4 2</td>
<td>305</td>
<td>0</td>
<td>1591</td>
<td>11</td>
<td>7 1,2,4,5,6,8,10,11,12</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

(22 clusters built)

For the best solution above, $s = 1.2e+04$.

### Experiments with k=3, criterion=fc

The 2 best clusterings follow... (6003 ms)

<table>
<thead>
<tr>
<th>Iter/Cluster</th>
<th>Seed</th>
<th>Except</th>
<th>Papers</th>
<th>Sim</th>
<th>Events Covered</th>
</tr>
</thead>
<tbody>
<tr>
<td>6 1</td>
<td>324</td>
<td>0</td>
<td>61</td>
<td>2</td>
<td>8 1.2,3,7.9.10</td>
</tr>
<tr>
<td>6 2</td>
<td>326</td>
<td>0</td>
<td>264</td>
<td>9</td>
<td>8 4.5,6.8,11.12</td>
</tr>
<tr>
<td>6 3</td>
<td>327</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td>5 7</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>5 1</td>
<td>329</td>
<td>0</td>
<td>91</td>
<td>9</td>
<td>8 1,2,3,9.10</td>
</tr>
<tr>
<td>5 2</td>
<td>330</td>
<td>0</td>
<td>354</td>
<td>5</td>
<td>7 4.5,6.8,11.12</td>
</tr>
<tr>
<td>5 3</td>
<td>331</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>5 7</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>

(76 clusters built)

For the best solution above, $s = 9.8e+03$.

### Experiments with k=4, criterion=fc

The 2 best clusterings follow... (76617 ms)

<table>
<thead>
<tr>
<th>Iter/Cluster</th>
<th>Seed</th>
<th>Except</th>
<th>Papers</th>
<th>Sim</th>
<th>Events Covered</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 1</td>
<td>334</td>
<td>0</td>
<td>72</td>
<td>2</td>
<td>7 1.2,9.10</td>
</tr>
<tr>
<td>3 2</td>
<td>335</td>
<td>0</td>
<td>176</td>
<td>3</td>
<td>7 3.7</td>
</tr>
<tr>
<td>3 3</td>
<td>336</td>
<td>0</td>
<td>33</td>
<td>4</td>
<td>7 4.11,12</td>
</tr>
<tr>
<td>3 4</td>
<td>337</td>
<td>0</td>
<td>33</td>
<td>6</td>
<td>7 6,8</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>2 1</td>
<td>338</td>
<td>0</td>
<td>15</td>
<td>1</td>
<td>6 1.2,10</td>
</tr>
<tr>
<td>2 2</td>
<td>339</td>
<td>0</td>
<td>477</td>
<td>9</td>
<td>7 3.7,9</td>
</tr>
<tr>
<td>2 3</td>
<td>340</td>
<td>0</td>
<td>33</td>
<td>12</td>
<td>7 4.11,12</td>
</tr>
<tr>
<td>2 4</td>
<td>341</td>
<td>0</td>
<td>33</td>
<td>6</td>
<td>7 6,8</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>(89 clusters built)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

For the best solution above, $s = 1.8e+04$.

With $beta = 3.00$ the best clustering at this level is for $k=3$.

### Experiments with $k=3$, criterion=fc

The 2 best clusterings follow... (13263 ms)

<table>
<thead>
<tr>
<th>Iter/Cluster</th>
<th>Seed</th>
<th>Except</th>
<th>Papers</th>
<th>Sim</th>
<th>Events Covered</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 1</td>
<td>283</td>
<td>0</td>
<td>91</td>
<td>2</td>
<td>8 1.2,3,7.9,10</td>
</tr>
<tr>
<td>5 2</td>
<td>302</td>
<td>0</td>
<td>294</td>
<td>12</td>
<td>8 4.5,6.8,11.12</td>
</tr>
<tr>
<td>5 3</td>
<td>305</td>
<td>0</td>
<td>0</td>
<td>7</td>
<td>5 7</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>

(22 clusters built)

For the best solution above, $s = 1.2e+04$.

With $beta = 3.00$ the best clustering at this level is for $k=3$.
beginning classification below hierarchy path 1-0

panic finding new seeds
experiment 2: k=3, criterion=fc

panic finding new seeds
premature end of clustering: seeds exhausted

the 2 best clusterings follow... (8887 ms)

<table>
<thead>
<tr>
<th>iter/splz</th>
<th>vl-rule</th>
<th>seed</th>
<th>except</th>
<th>parents</th>
<th>min</th>
<th>events covered</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>[mp=6520][ran=14][mov=10][disp=color;tv][keys=52, 53]</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5 1,2</td>
</tr>
<tr>
<td>2</td>
<td>[mp=6520][ran=14][mov=10][disp=color;tv][keys=52, 53]</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5 3</td>
</tr>
<tr>
<td>totals</td>
<td></td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>10 1,2,3,5,10</td>
</tr>
</tbody>
</table>

(19 stars built)

for the best solution above, n = 2.9e+02

panic finding new seeds
experiment 2: k=3, criterion=fc

panic finding new seeds
premature end of clustering: seeds exhausted

the 2 best clusterings follow... (8887 ms)

<table>
<thead>
<tr>
<th>iter/splz</th>
<th>vl-rule</th>
<th>seed</th>
<th>except</th>
<th>parents</th>
<th>min</th>
<th>events covered</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>[mp=6520][ran=14][mov=10][disp=color;tv][keys=52, 53]</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5 1,2</td>
</tr>
<tr>
<td>2</td>
<td>[mp=6520][ran=14][mov=10][disp=color;tv][keys=52, 53]</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5 1,2</td>
</tr>
<tr>
<td>3</td>
<td>[mp=6520][ran=14][mov=10][disp=color;tv][keys=52, 53]</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5 3</td>
</tr>
<tr>
<td>totals</td>
<td></td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>12 1,2,3,5,10</td>
</tr>
</tbody>
</table>

(21 stars built)

for the best solution above, n = 1.4e+02

experiment 2: k=4, criterion=fc

panic finding new seeds
premature end of clustering: seeds exhausted

the 1 best clusterings follow... (7510 ms)

<table>
<thead>
<tr>
<th>iter/splz</th>
<th>vl-rule</th>
<th>seed</th>
<th>except</th>
<th>parents</th>
<th>min</th>
<th>events covered</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>[mp=6520][ran=14][mov=10][disp=color;tv][keys=52, 53]</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5 1</td>
</tr>
<tr>
<td>2</td>
<td>[mp=6520][ran=14][mov=10][disp=color;tv][keys=52, 53]</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5 2</td>
</tr>
<tr>
<td>3</td>
<td>[mp=6520][ran=14][mov=10][disp=color;tv][keys=52, 53]</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5 3</td>
</tr>
<tr>
<td>4</td>
<td>[mp=6520][ran=14][mov=10][disp=color;tv][keys=52, 53]</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5 9,10</td>
</tr>
<tr>
<td>totals</td>
<td></td>
<td>16</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>44 1,2,3,5,10</td>
</tr>
</tbody>
</table>

(30 stars built)

for the best solution above, n = 2.6e+02

with beta= 3.00 the best clustering at this level is for k=3

7
- 25 -

<table>
<thead>
<tr>
<th>iter/cpil#</th>
<th>Vi-rule</th>
<th>seed</th>
<th>except</th>
<th>ptpars</th>
<th>sim</th>
<th>events covered</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>[mp=680][ran=46k] [rom=4k..16k]</td>
<td>4</td>
<td>0</td>
<td>33</td>
<td>7 4.11.12</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>[mp=800c][ran=64k] [rom=4k..16k]</td>
<td>5</td>
<td>0</td>
<td>33</td>
<td>7 5.5.8</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>[mp=680][ran=64k] [rom=4k..16k]</td>
<td>5</td>
<td>0</td>
<td>21</td>
<td>8 6.5.12</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>[mp=680][ran=64k] [rom=4k..16k]</td>
<td>5</td>
<td>0</td>
<td>27</td>
<td>20</td>
</tr>
</tbody>
</table>

Experiment 2: k=4, criterion=rc

The best clustering follows... (8900 ms)

<table>
<thead>
<tr>
<th>iter/cpil#</th>
<th>Vi-rule</th>
<th>seed</th>
<th>except</th>
<th>ptpars</th>
<th>sim</th>
<th>events covered</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>[mp=680][ran=46k] [rom=4k..16k]</td>
<td>4</td>
<td>0</td>
<td>33</td>
<td>7 4.11.12</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>[mp=800c][ran=64k] [rom=4k..16k]</td>
<td>5</td>
<td>0</td>
<td>33</td>
<td>7 5.5.8</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>[mp=680][ran=64k] [rom=4k..16k]</td>
<td>5</td>
<td>0</td>
<td>21</td>
<td>8 6.5.12</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>[mp=680][ran=64k] [rom=4k..16k]</td>
<td>5</td>
<td>0</td>
<td>27</td>
<td>20</td>
</tr>
</tbody>
</table>

Experiment 2: k=4, criterion=rc

The 2 best clusterings follow... (14200 ms)

<table>
<thead>
<tr>
<th>iter/cpil#</th>
<th>Vi-rule</th>
<th>seed</th>
<th>except</th>
<th>ptpars</th>
<th>sim</th>
<th>events covered</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>[mp=680][ran=46k] [rom=4k..16k]</td>
<td>4</td>
<td>0</td>
<td>33</td>
<td>7 4.11.12</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>[mp=800c][ran=64k] [rom=4k..16k]</td>
<td>5</td>
<td>0</td>
<td>33</td>
<td>7 5.5.8</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>[mp=680][ran=64k] [rom=4k..16k]</td>
<td>5</td>
<td>0</td>
<td>21</td>
<td>8 6.5.12</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>[mp=680][ran=64k] [rom=4k..16k]</td>
<td>5</td>
<td>0</td>
<td>27</td>
<td>20</td>
</tr>
</tbody>
</table>

Experiment 2: k=4, criterion=rc

Picking the 4 seeds:

The 2 best clusterings follow... (10017 ms)

<table>
<thead>
<tr>
<th>iter/cpil#</th>
<th>Vi-rule</th>
<th>seed</th>
<th>except</th>
<th>ptpars</th>
<th>sim</th>
<th>events covered</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>[mp=680][ran=46k] [rom=4k..16k]</td>
<td>4</td>
<td>0</td>
<td>6</td>
<td>7 4.11.12</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>[mp=800c][ran=64k] [rom=4k..16k]</td>
<td>5</td>
<td>0</td>
<td>6</td>
<td>7 4.11.12</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>[mp=680][ran=64k] [rom=4k..16k]</td>
<td>5</td>
<td>0</td>
<td>4</td>
<td>7 6.5.12</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>[mp=680][ran=64k] [rom=4k..16k]</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>5 8</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>[mp=680][ran=64k] [rom=4k..16k]</td>
<td>5</td>
<td>0</td>
<td>6</td>
<td>7 4.11.12</td>
</tr>
</tbody>
</table>

For the best solution above, k= 5.1e+02

With beta = 3.00 the best clustering at this level is for k=4

<table>
<thead>
<tr>
<th>iter/cpil#</th>
<th>Vi-rule</th>
<th>seed</th>
<th>except</th>
<th>ptpars</th>
<th>sim</th>
<th>events covered</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>[mp=680][ran=46k] [rom=4k..16k]</td>
<td>4</td>
<td>0</td>
<td>6</td>
<td>7 4.11.12</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>[mp=800c][ran=64k] [rom=4k..16k]</td>
<td>5</td>
<td>0</td>
<td>6</td>
<td>7 4.11.12</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>[mp=680][ran=64k] [rom=4k..16k]</td>
<td>5</td>
<td>0</td>
<td>4</td>
<td>7 6.5.12</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>[mp=680][ran=64k] [rom=4k..16k]</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>5 8</td>
</tr>
</tbody>
</table>
REFERENCES


Appendix I: CLUSTER/2 Program Listing

Index to procedures and functions

<table>
<thead>
<tr>
<th>page</th>
<th>name</th>
<th>page</th>
<th>name</th>
<th>page</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>72</td>
<td>procedure addehr</td>
<td>67</td>
<td>function genpath</td>
<td>66</td>
<td>procedure prtastr</td>
</tr>
<tr>
<td>72</td>
<td>procedure addnum</td>
<td>42</td>
<td>procedure genrliz</td>
<td>50</td>
<td>function prval</td>
</tr>
<tr>
<td>72</td>
<td>procedure addval</td>
<td>23</td>
<td>procedure getinput</td>
<td>24</td>
<td>function readcolumn</td>
</tr>
<tr>
<td>45</td>
<td>function bestc</td>
<td>81</td>
<td>procedure hlevel</td>
<td>58</td>
<td>procedure reduce</td>
</tr>
<tr>
<td>13</td>
<td>function condist</td>
<td>61</td>
<td>function iabs</td>
<td>41</td>
<td>function refresh</td>
</tr>
<tr>
<td>75</td>
<td>procedure clearcv</td>
<td>43</td>
<td>function intract</td>
<td>41</td>
<td>function reflev</td>
</tr>
<tr>
<td>78</td>
<td>function clstring</td>
<td>66</td>
<td>procedure mane</td>
<td>40</td>
<td>function refshow</td>
</tr>
<tr>
<td>67</td>
<td>procedure cluster</td>
<td>61</td>
<td>function maplow</td>
<td>40</td>
<td>function refnum</td>
</tr>
<tr>
<td>40</td>
<td>function cmpecov</td>
<td>60</td>
<td>procedure masgn</td>
<td>43</td>
<td>procedure refusion</td>
</tr>
<tr>
<td>29</td>
<td>procedure cripdef</td>
<td>59</td>
<td>function mcard</td>
<td>70</td>
<td>function savecv</td>
</tr>
<tr>
<td>18</td>
<td>procedure critval</td>
<td>60</td>
<td>function mcomp</td>
<td>29</td>
<td>procedure semantics</td>
</tr>
<tr>
<td>45</td>
<td>function cselect</td>
<td>60</td>
<td>function min</td>
<td>59</td>
<td>procedure setlevelmap</td>
</tr>
<tr>
<td>44</td>
<td>function degisct</td>
<td>57</td>
<td>procedure mop</td>
<td>58</td>
<td>procedure setmap</td>
</tr>
<tr>
<td>36</td>
<td>procedure dh</td>
<td>48</td>
<td>function newcplx</td>
<td>21</td>
<td>procedure setup</td>
</tr>
<tr>
<td>73</td>
<td>procedure dispart</td>
<td>62</td>
<td>procedure nid</td>
<td>23</td>
<td>function skipfill</td>
</tr>
<tr>
<td>71</td>
<td>procedure disprule</td>
<td>49</td>
<td>function numcplx</td>
<td>11</td>
<td>function star</td>
</tr>
<tr>
<td>49</td>
<td>function domof</td>
<td>59</td>
<td>function numsel</td>
<td>13</td>
<td>procedure starcr</td>
</tr>
<tr>
<td>73</td>
<td>procedure dptrbufs</td>
<td>13</td>
<td>function numstar</td>
<td>61</td>
<td>procedure statnam</td>
</tr>
<tr>
<td>36</td>
<td>procedure dt</td>
<td>51</td>
<td>procedure printreal</td>
<td>11</td>
<td>procedure strasgn</td>
</tr>
<tr>
<td>28</td>
<td>function dtypval</td>
<td>52</td>
<td>procedure prt</td>
<td>21</td>
<td>function stringmatch</td>
</tr>
<tr>
<td>63</td>
<td>function elim</td>
<td>50</td>
<td>function prtalfa</td>
<td>47</td>
<td>function syndist</td>
</tr>
<tr>
<td>23</td>
<td>function elininput</td>
<td>8</td>
<td>procedure prtbmsg</td>
<td>26</td>
<td>procedure tablesetup</td>
</tr>
<tr>
<td>23</td>
<td>procedure errormsg</td>
<td>51</td>
<td>procedure prtcrit</td>
<td>39</td>
<td>function tcover</td>
</tr>
<tr>
<td>44</td>
<td>procedure extend</td>
<td>52</td>
<td>procedure prtmap</td>
<td>47</td>
<td>procedure trim</td>
</tr>
<tr>
<td>49</td>
<td>function freecplx</td>
<td>53</td>
<td>procedure prtrit</td>
<td>22</td>
<td>function typcod</td>
</tr>
<tr>
<td>42</td>
<td>procedure genbest</td>
<td>51</td>
<td>procedure prset</td>
<td>61</td>
<td>function wmcard</td>
</tr>
</tbody>
</table>
procedure stack: (** save star tracing criterion spec**) c (ct); percolate; (** save star tracing criterion spec**) c (ct);

begin if sentinel then new(pascal); pascal-staccount := 0; pascal-closed := false; end;

function number: (** report the number of stars built since last repeat**) c (ct); integer; begin pascal-stacount := pascal-stacount + 1; end;

function condition: (** select new seeds - central or distant**) c (ct); boolean; begin c (ct); false; (** new seeds are selected by measuring the sum of synaptic distance from each seed to all others. central seeds have a more symmetric distance distribution. distant seeds are reserved more rapidly. the set of n new events that do not duplicate any previously used seed set, the seed set is compared with the previously generated seed set and if there is a match, the function end; end; end; end; end; end; end;

if accept? then begin

begin if seed? then begin

end;

end;

end;

end;

end;

end;
end;
28: begin
  (prop structure)
  refer : all;
  inserted := 0;
  strmatch Venezia;
end;
29: begin
  (prop name)
  name := inchar afirma[1];
  refer := strref structure;
  ismatch := strmatch refer;
  if refer/name do begin
    if (name = refer/name) then begin
      inserted := inserted + 1;
      refer/math := refer/math + 1;
    end;
    if refer/name do begin
      refer := refer + 1;
    end;
  end;
end;
30: if refer/val = true then begin
  ismatch := refer/val = true;
  refer := refer + 1;
end;
end;
31: begin
  (prop value)
  (struct subspace)
  name := inchar afirma[1];
  send := strref structure;
  ismatch := refer/name;
  while (refer/name) do begin
    if name = refer/name then begin
      refer := refer + 1;
      send := send + 1;
      ismatch := ismatch + 1;
    end;
  end;
end;
32: begin
  (event base value)
  refer := refer + 1;
  if refer/name do begin
    if refer/name do begin
      refer := refer + 1;
    end;
  end;
end;
33: begin
  (event base value)
  refer := refer + 1;
  if refer/name do begin
    if refer/name do begin
      refer := refer + 1;
    end;
  end;
end;
34: begin
  (event base value)
  refer := refer + 1;
  if refer/name do begin
    if refer/name do begin
      refer := refer + 1;
    end;
  end;
end;
35: begin
  (prop crit)
  if refer/value then begin
    inserted := inserted + 1;
    refer/prec := refer/prec + 1;
  end;
end;
36: begin
  (prop crit)
  if refer/value then begin
    inserted := inserted + 1;
    refer/prec := refer/prec + 1;
  end;
end;
37: begin
  (event left value)
  if refer/value then begin
    if refer/value then begin
      refer := refer + 1;
    end;
  end;
end;
38: begin
  (print parameter)
  for refer : to refer do begin
    if refer/value then begin
      refer/value := refer/value + 1;
      refer := refer + 1;
      strmatch Venezia;
    end;
    if refer/value then begin
      refer/value := refer/value + 1;
      refer := refer + 1;
    end;
    if refer/value then begin
      refer/value := refer/value + 1;
      refer := refer + 1;
    end;
    if refer/value then begin
      refer/value := refer/value + 1;
      refer := refer + 1;
    end;
  end;
end;
39: begin
  (print parameter)
  for refer : to refer do begin
    if refer/value then begin
      refer/value := refer/value + 1;
      refer := refer + 1;
    end;
    if refer/value then begin
      refer/value := refer/value + 1;
      refer := refer + 1;
    end;
    if refer/value then begin
      refer/value := refer/value + 1;
      refer := refer + 1;
    end;
    if refer/value then begin
      refer/value := refer/value + 1;
      refer := refer + 1;
    end;
  end;
end;
40: begin
  (define header keyord)
  (* the format is: 63('word', 'kid, art, co') *)
  (* where kid is a table of number, act in a semantic action number *)
  (* and conv is a input conversion code, or "natural" if nothing is read *)
begin
  in := stdin;
  v_input := stdin;
  table := stdin;
  conv := stdin;
begin
  (* define relational tables *)
  (* for format: st('handle name', .st)) *)
begin
  (define relational table)
  (* for format: st('handle name', .st) *)
begin
  (define relational table)
  (* for format: st('handle name', .st) *)
begin
  (* typecodes below denote character classes: *)
  (p80_chars, (p80_c0, q80, e80, i80, d80, a80, e80, i80, d80, a80));
begin

procedure get_cluster_vertices(!c: integer!); begin
  case c of
    start: begin
      // code
    end;
end;

function get_next_vertex(!v: integer!); begin
  case v of
    0: begin
      // code
    end;
end;

function get_next_cluster(!v: integer!); begin
  case v of
    0: begin
      // code
    end;
end;

begin
  // code
end;

procedure get_clusters(!c: integer!); begin
  case c of
    start: begin
      // code
    end;
end;

begin
  // code
end;

procedure get_vertices(!c: integer!); begin
  case c of
    start: begin
      // code
    end;
end;

function get_next_vertex(!v: integer!); begin
  case v of
    0: begin
      // code
    end;
end;

function get_next_cluster(!v: integer!); begin
  case v of
    0: begin
      // code
    end;
end;

begin
  // code
end;

procedure get_clusters(!c: integer!); begin
  case c of
    start: begin
      // code
    end;
end;

begin
  // code
end;

procedure get_vertices(!c: integer!); begin
  case c of
    start: begin
      // code
    end;
end;

function get_next_vertex(!v: integer!); begin
  case v of
    0: begin
      // code
    end;
end;

function get_next_cluster(!v: integer!); begin
  case v of
    0: begin
      // code
    end;
end;

begin
  // code
end;

procedure get_clusters(!c: integer!); begin
  case c of
    start: begin
      // code
    end;
end;

begin
  // code
end;

procedure get_vertices(!c: integer!); begin
  case c of
    start: begin
      // code
    end;
end;

function get_next_vertex(!v: integer!); begin
  case v of
    0: begin
      // code
    end;
end;

function get_next_cluster(!v: integer!); begin
  case v of
    0: begin
      // code
    end;
end;

begin
  // code
end;

procedure get_clusters(!c: integer!); begin
  case c of
    start: begin
      // code
    end;
end;

begin
  // code
end;

procedure get_vertices(!c: integer!); begin
  case c of
    start: begin
      // code
    end;
end;

function get_next_vertex(!v: integer!); begin
  case v of
    0: begin
      // code
    end;
end;

function get_next_cluster(!v: integer!); begin
  case v of
    0: begin
      // code
    end;
end;

begin
  // code
end;

procedure get_clusters(!c: integer!); begin
  case c of
    start: begin
      // code
    end;
end;

begin
  // code
end;

procedure get_vertices(!c: integer!); begin
  case c of
    start: begin
      // code
    end;
end;

function get_next_vertex(!v: integer!); begin
  case v of
    0: begin
      // code
    end;
end;

function get_next_cluster(!v: integer!); begin
  case v of
    0: begin
      // code
    end;
end;

begin
  // code
end;
if any path is evolved beyond a certain size, it is considered as a "path". 

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.

end.
null
REFERENCES


REFERENCES


