Tabular Representation of Relations

David Lorge Parnas
Telecommunications Research Institute of Ontario
Communications Research Laboratory
Department of Electrical and Computer Engineering
McMaster University, Hamilton, Ontario Canada L8S 4K1

ABSTRACT

Multi-dimensional mathematical expressions, called tables, have proven to be useful for documenting digital systems. This paper describes 10 classes of tables, giving their syntax and semantics. Several abbreviations that can be useful in tables are introduced. Simple examples are provided.

1 Introduction

In earlier papers, [1,2], we have shown how the documentation required for the professional construction and use of computing systems can consist of descriptions of a set of mathematical relations. Those papers discuss the documents very abstractly; the contents of the documents are specified without restricting the notations or formats to be used. This paper complements the earlier papers by defining multidimensional notations (which we call tabular expressions) that have proven useful for describing the specified mathematical functions in practical applications [3, 4, 5, 6, 7, 8]. A companion paper [9], presents an interpretation of logical expressions that is designed for these computer engineering applications. These logical expressions, together with conventional mathematical formulae, are the components of the tabular expressions. The notations defined in this paper are intended to supplement, not replace, notation that is already in use by engineers.

To understand the purpose of this paper, one must recall the distinction between a function, its description, and a practical means of computing its values. The function is an abstraction that may be thought of as a set of ordered pairs; the description is something we can write on paper (or some other medium) to describe that function; a procedure to be carried out using a specified piece of hardware will be the mechanism that is expected to compute values of the function for specified values in its domain.

For example, each of the expressions below describe a function whose domain and range are the set of real numbers. The expressions look very different, but the meaning (i.e. the function) is the same.

$$F(x) = x + 1$$

$$\lambda(y)\ [y + 1]$$

$$F(x) = (x = 1 \rightarrow 2,\ x \neq 1 \rightarrow x + 1)$$

$$\{(x,y) \mid y = x + 1\}$$

A wide variety of mechanisms could compute values of the function described above.

Such notational variations exist because no single notation is well suited for describing all mathematical functions and relations. The history of mathematics and engineering shows clearly

that when we become interested in a new class of functions, we also invent new notations that are well-suited for the description of functions in that class. The functions that arise in the description of computer systems have two important characteristics. First, digital technology allows us to implement functions that have many discontinuities, discontinuities that can occur at arbitrary points in the domain of the function. Unlike the designers of analogue systems, we are not constrained to implement functions that are either continuous or exhibit exploitable regularity in their discontinuities. Second, the range and domain of these functions are often tuples whose elements are of distinct types and the tuples' values cannot be described in terms of a typical element. The use of traditional mathematical notation, developed for functions that do not have these characteristics, often results in function descriptions that are complex, lengthy, and hard to read. As a consequence of this complexity, mathematical specifications are often incorrect, and, even more often, misunderstood. This has led many people to conclude that it is not practical to provide precise mathematical descriptions of computer systems.

In his work on functional semantics, Mills and his colleagues [e.g. 10] have used the "concurrent-assignment statement" to represent functions whose range and domain are tuples whose elements are identified by variables. For example, the expression below can be interpreted as describing a function whose domain comprises triples, (x,y,z) and whose range comprises pairs [10]. This notation appeals to programmers because it closely resembles familiar programming language notation for assignment.

$$x,y := x + y, z + 1$$

While the concurrent-assignment notation is a contribution to the description of such functions, and is useful for small examples, we have not found it suitable for the description of the functions that arise in practical computer system design. The use of conditional expressions in concurrent-assignment statements can result in a lengthy text that is difficult to parse and, consequently, difficult to check or use. If the relation we want to describe is not a function, concurrent-assignment notation does not work without further embellishment.

Conventional mathematical notation is essentially 1-dimensional. Although some use is made of the second dimension, e.g. by raising superscripts, lowering subscripts and using summation symbols that are larger than the other characters, such notation is easily converted to a one dimensional string of symbols with little loss of readability. In practical experience that began in 1977 [5, 7, 3], we have found that truly multidimensional expressions, intuitively described as tabular expressions, can be of great practical value when describing the functions and relations that arise in computer system design. The multidimensional organisation of large amounts of information eases the task of finding the small amount of information that is needed to answer particular questions and supports a "divide and conquer" approach to understanding a complex description.

Many have used these tabular expressions on an *ad hoc* basis, relying on the intuitive understanding that comes from the similarity of these tables to tables that we encounter in our daily lives. However, widespread use, and mechanical support, of formal tabular notations will require precise definitions of their meaning. This paper proposes such definitions for ten varieties of tabular expressions.

2 The Constituents of Tabular Expressions (Tables)

Like the terms and expressions defined in [9], tables are constructed recursively from simpler components, namely conventional expressions and grids. Below we describe how tables may be constructed from these components. Section 3 of this paper describes a variety of ways in which tables may be interpreted as predicates and functions.

2.1 Scalar Expressions

A term or predicate expression as defined in [9] will be called a *scalar expression.* Functions and relations that have been described using the notation of this paper can be given names and those names can be included in the sets of function-names and relation-names to be used subsequently to form scalar expressions. Nothing else is a scalar expression.

2.2 Grids

A *grid*, G, is an indexed set[1] such that, for positive integers dim(G), and $len_1(G)$, ..., $len_{dim(G)}(G)$,

- the index set of G, (a set of tuples), is the Cartesian product of the sets:

$$\{1,2, ..., len_1(G)\}, \{1, 2, ..., len_2(G)\}, ..., \{1, 2, ..., len_{dim(G)}(G)\},$$

- the elements of G are either expressions (cf. Section 2.4) or previously constructed grids.

For any grid, G:

- dim*(G)* is the *dimensionality,* i.e. the number of dimensions, of G,
- $len_i(G)$ is the *length* of the grid G in its i^{th} dimension,
- *shape(G)* is a tuple of length dim(G) whose i^{th} element is $len_i(G)$,
- G_I denotes the element of G with index I, where I is a member of the index set of G,
- two elements of G are said to be *distinct* if their indices are different,
- G is said to be *disjoint* if (a) all of its elements are predicate expressions, and (b) all conjunctions of distinct elements of that grid are equivalent to **false**,
- G is said to be *constant* if all its elements are identical,
- an *i-stripe* of G ($1 \leq i \leq dim(G)$) is a grid with shape = ($len_i(G)$) comprising elements of G whose indices differ only in their i^{th} element. If I is in the index set of G, $G^i_{I|i}$ denotes the i-stripe that includes G_I.[2] The j^{th} element of $G^i_{I|i}$, denoted $G^i_{I|i,\ j}$, is $G_{I|i|j}$. In 2-dimensional tables, a 1-stripe is called a *column* and a 2-stripe a *row.* We use the general term *stripe* to denote an i-stripe for any i.

[1] For our purposes, S is an i*ndexed set* if there exists a set, I, (the *index set),* and a 1:1 mapping, *identifies,* from I to S. If (i,s) ε identifies, we may write S_i for s. See [12] for more discussion.

[2] If I is a tuple, and j is not greater than the length of I, *"I|j"* denotes a tuple obtained by deleting the j^{th} element of I, *"I|j|k"* denotes a tuple obtained by replacing the j^{th} element of I with k, and *"I_j"* denotes the j^{th} element of I. Literal tuples will be enclosed in angle brackets (e.g "<2,1>"), which can be omitted for 1-tuples.

2.3 Tables

A table, T, consists of a main grid, G, and grids H_1, H_2, ..., $H_{dim(G)}$, (known as headers), such that for any i, $1 \leq i \leq dim(G)$, $shape(H_i) = shape(G)|i$.

For any table, T:

- *dim(T)* = dim(G),
- $len_i(T) = len_i(G)$,
- the *index set of T* is the index set of G,
- $T_I = G_I$,
- $T_{I,m}$ denotes the element of H_m with index I|m.,
- for 2-dimensional tables, H_1 and H_2 can be thought of as being the column and row headings respectively,
- *an i,j-headerstripe* of T ($1 \leq i \leq dim(T)$, $1 \leq j \leq dim(T)$, $i \neq j$) is a grid with shape = ($len_i(T)$) comprising elements of H_j whose indices differ only in their $r(i,j)^{th}$ element, where r(i,j) = i if i<j and i -1 otherwise,
- there are no i,j-headerstripes with i = j.
- If I is in the index set of T, $T^{i,j}_{I|i}$ denotes the i,j-headerstripe that includes $T_{I,j}$. The k^{th} element of $T^{i,j}_{I|i}$, denoted $T^{i,j}_{I|i,k}$, is $T_{I|i|k,j}$,
- we use the term *i-headerstripe* for any i,j-headerstripe.

2.4 Expressions

Any term as defined in [9], and any table that is to be interpreted as a function, is a *term*. Any predicate expression as defined in [9], and any table form that is to be interpreted as a predicate is a *predicate expression*. Any term or predicate expression is an *expression*. These expressions may be included in larger expressions, (in accordance with [9]), or appear as elements of grids.

3 The Semantics of Tables

The above describes the syntax of tables, but not their meaning. We have encountered a variety of interpretations of tables that appear to be useful in specific circumstances. In the following section we will describe these tables, first by giving further details about their syntax, then by describing the relation that they represent. In use, each table must be tagged to indicate the intended interpretation. The various interpretations do not have enough in common to justify attempting a single, more general, definition of all interpretations.

3.1 Normal Function Tables

A *normal function table*, T, is a table in which the elements of the main grid, G, are terms and the elements of the headers are predicate expressions. T will be interpreted as a function whose domain is a set of assignments [9].[3]

[3] An a*ssignment* is a tuple containing one element for every variable that may appear in expressions. As in [9], we assume that the set of variables is finite.

A normal function table, T, is said to be *proper* if and only if, for any indices I, J of T, the conjunction of $T_{I,1}$, $T_{I,2}$, ..., $T_{I,dim(T)}$, $T_{J,1}$, $T_{J,2}$, ..., $T_{J,dim(T)}$ is equivalent to **false** when $I \neq J$.

An index of T, I, can be associated with a function F_I as described below.

- The domain of F_I is the set of assignments characterised by the conjunction of $T_{I,1}$, $T_{I,2}$, ..., $T_{I,dim(T)}$.
- Within that domain, the value of F_I is obtained by evaluating the term G_I.

A proper normal function table, T, can be interpreted as a function, F_T, that is the union[4] of the functions F_I, for all I in the index set of T.

	y = 27	y > 27	y < 27
			H_1
x = 3	$27 + \sqrt{27}$	$54 + \sqrt{27}$	$y^2 + 3$
x < 3	$27 + \sqrt{-(x-3)}$	$y + \sqrt{-(x-3)}$	$y^2 + (x-3)^2$
x > 3	$27 + \sqrt{x-3}$	$2 \times y + \sqrt{x-3}$	$y^2 + (3-x)^2$
H_2			**G**

Figure 1: A Normal Function Table

3.2 Inverted Function Tables

An *inverted function table*, T, is a table in which the elements of the main grid, G, are predicate expressions, the elements of H_2, ..., $H_{dim(T)}$ are predicate expressions, and the elements of H_1 are terms. T will be interpreted as a function whose domain is a set of assignments.

An inverted function table,T, is said to be *proper* if and only if, for any indices I, J of T, the conjunction of $T_{I,2}$, $T_{I,3}$, ..., $T_{I,dim(T)}$, $T_{J,2}$, $T_{J,3}$, ..., $T_{J,dim(T)}$, G_I, G_J is equivalent to **false** if $I \neq J$.

An index, I, in the index set of T can be associated with a function F_I as described below.

- Let K_I be the conjunction of $T_{I,2}$, ..., $T_{I,dim(T)}$. The domain of F_I is the set of assignments characterised by the conjunction of K_I and G_I
- Within that domain, the value of F_I is obtained by evaluating the term $T_{I,1}$.

A proper inverted function table, T, can be interpreted as a function, F_T, that is the union of the functions F_I for all I in the index set of T.

[4] Since we view functions as sets of pairs, the union of functions is set-union.

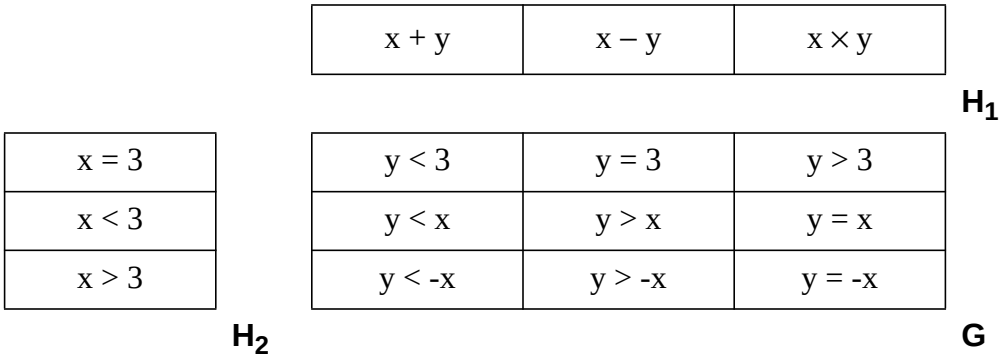


Figure 2: An Inverted Function Table

3.3 Vector Function Tables [5]

A *vector function table, T,* is a table in which the elements of the main grid, G, are terms, the elements of H_1, H_3, ..., $H_{dim(T)}$[6] are predicate expressions, and the elements of H_2 are single variables. These tables describe a function whose range is a set of $len_1(T)$-tuples and whose domain is a set of assignments. The elements of H_2 are used to associate individual elements of the $len_1(T)$-tuples in the range with elements of the assignments in the domain of the function.

A vector function table,T, is said to be *proper* if and only if:

- for any indices I, J of T, the conjunction of $T_{I,1}$, $T_{I,3}$, $T_{I,4}$..., $T_{I,dim(T)}$, $T_{J,1}$, $T_{J,3}$, $T_{J,4}$ ..., $T_{J,dim(T)}$ is equivalent to **false** when $I \neq J$.
- In H_2, all stripes other than 1-stripes are constant.
- No two elements of a 1-stripe of H_2 are the same variable.

Let I be in the index set of T. A function $F_{I|1}$ can be associated with I|1 as follows

- The domain of $F_{I|1}$ is the set of assignments that is characterised by the conjunction of $T_{I,1}$, $T_{I,3}$, $T_{I,4}$, ..., $T_{I,dim(T)}$.
- Within that domain, the value of $F_{I|1}$ is a tuple of length $len_1(G)$. For $j \leq len_1(G)$, the value of the j^{th} element of the tuple is determined by evaluating $G_{I|1|j}$.

A proper vector function table, T, can be interpreted as a Function, F_T, that is the union of all the functions $F_{I|1}$ for all I in the index set of T.

[5] Strictly speaking we should use “tuple” instead of “vector”, but this usage has become common.

[6] If dim(T) < 3, the sequence H_3, ..., $H_{dim(T)}$ is empty; if dim(T) =3, the sequence H_3, ..., $H_{dim(T)}$ is H_3.

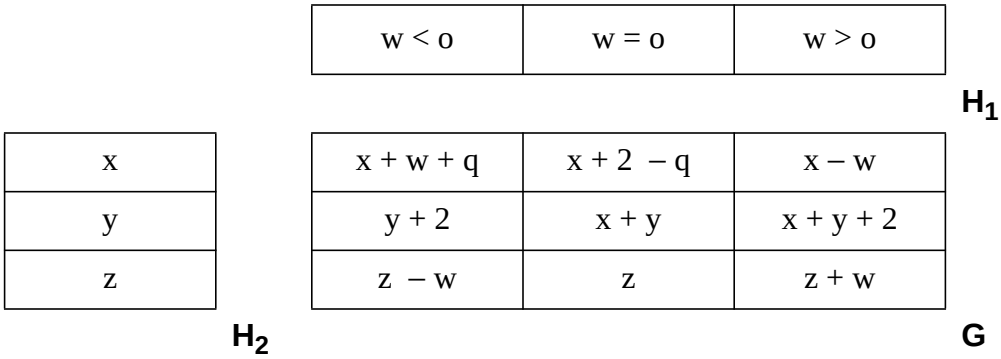


Figure 3: A Vector Function Table

3.4 Normal Relation Tables

A *normal relation table*, T, is a table in which the elements of the main grid and headers are predicate expressions. The expressions in the main grid and headers are constructed from the usual variables except that one variable, which will be written, "®", may not appear in the headers. The relations represented by these tables will be viewed as a set of pairs of the form (a, y) where a is an assignment ([9]) and y is a scalar element of the Universe.

A normal relation table, T, is said to be *proper* if and only if for any indices I, J of T, the conjunction of $T_{I,1}$, $T_{I,2}$, ..., $T_{I,dim(T)}$, $T_{J,1}$, $T_{J,2}$, ..., $T_{J,dim(T)}$ is equivalent to **false** when $I \neq J$.

A member of the index set of T, I, can be associated with a relation R_I as described below.

- Let a and a' be assignments.
- A pair (a, y) is in R_I, if and only if

  (1) a is in the set of assignments characterised by the conjunction of $T_{I,1}$, $T_{I,2,}$, ..., $T_{I,dim(T)}$, and

  (2) the set of assignments characterised by G_I contains the assignment a' obtained from a by replacing the value of the element corresponding to ® with y.

A proper normal relation table, T, can be interpreted as a relation, R_T, that is the union of all the relations R_I for all I in the index set of T.

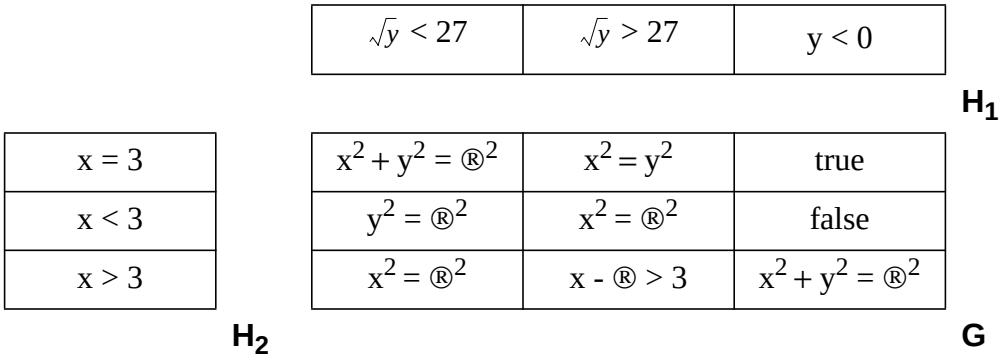


Figure 4: A Normal Relation Table

3.5 Inverted Relation Tables

An *inverted relation table,* T, is a table in which the elements of the main grid and headers are predicate expressions. The expressions in H_2, ..., $H_{dim(T)}$ and G are constructed using the usual variables except that one variable, which will be written "®", may not appear. The expressions in H_1 may include "®". The relations represented by these tables will be viewed as a set of pairs of the form (a, y) where a is an assignment and y is a scalar element of the Universe.

An inverted relation table,T, is said to be *proper* if and only if, for any indices I, J of T, the conjunction of $T_{I,2}$, $T_{I,3}$, ..., $T_{I,dim(T)}$, $T_{J,2}$, $T_{J,3}$, ..., $T_{J,dim(T)}$, G_I, G_J is equivalent to **false** if $I \neq J$.

An index of T, I, can be associated with a relation, R_I, as described below.

- Let K_I be the conjunction of $T_{I,2, ..., } T_{I,dim(T)}$.
- Let a and a' be assignments.
- A pair (a, y) is in R_I, if and only if:

  (1) a is in the set of assignments characterised by the conjunction of K_I with G_I, and

  (2) the set of assignments characterised by $T_{I,1}$ contains the assignment a' obtained from a by replacing the value of the element corresponding to ® with y.

A proper inverted relation table, T, can be interpreted as a relation, R_T, that is the union of the relations R_I for all I in the index set of T.

	$x^2 + y^2 = ®^2$	® = 3	$x^2 + ®^2 = y^2$
			H_1
x = 3	y > 3	y = 3	y < 3
x < 3	y < 0	y ≥ 0	false
x > 3	y > 100	y = 100	y < 100
H_2			**G**

Figure 5: An Inverted Relation Table

3.6 Vector Relation Tables

A *vector relation table, T,* is a table in which the elements of the main grid, G, are predicate expressions, the elements of H_1, H_3, ..., $H_{dim(T)}$ are predicate expressions, and the elements of H_2 are single variables. T will be interpreted as a relation whose domain is a set of assignments and range is a set of $len_1(T)$-tuples. The variables that are the elements of H_2 are used to associate elements of the $len_1(T)$-tuples in the range with elements of the assignments in the domain of the relation.

A vector relation table, T, is said to be *proper* if and only if:

- for any indices I, J of T, the conjunction of $T_{I,1}$, $T_{I,3}$, $T_{I,4}$, ..., $T_{I,dim(T)}$, $T_{J,1}$, $T_{J,3}$, $T_{J,4}$, ..., $T_{J,dim(T)}$ is equivalent to **false** when $I \neq J$.
- In H_2, all stripes other than 1-stripes are constant.

- No two elements of a 1-stripe of H_2 are the same variable.

Let I be in the index set of T. A relation $R_{I|1}$ can be associated with I|1 as follows

- Let K_I be the conjunction of $T_{I,1}$, $T_{I,3}$, $T_{I,4}$, ..., $T_{I,dim(T)}$.
- Let a and a' be assignments. Let V be a tuple with length = $len_1(G)$. V_j will be associated with the variable that is the j^{th} element of the 1-stripes of H_2.
- A pair (a, V) is in $R_{I|1}$, if and only if

  (1) a is in the set of assignments characterised by K_I, and

  (2) the set of assignments characterised by the conjunction of $G_{I|1|1}$, ..., $G_{I|1|n}$, where n = $len_1(G)$, contains an assignment a' that is identical to a in all positions except (possibly) those corresponding to the variables mentioned in H_2, and the elements of a' corresponding to variables mentioned in H_2 are the values of the elements of V associated with the same variable. [7]

A proper vector relation table, T, can be interpreted as a relation, R_T, that is the union of the relations $R_{I|1}$ for all I in the index set of T.

	$w < o$	$w = o$	$w > o$
			H_1
x	$x = w$	$x^2 = 4$	$(x)^2 = w$
y	$y^2 = x + 2$	$y = x + 2$	$y = \|x\| + 2$
z	$z^2 = x^2 + y^2 + w^2$	$z^2 = x^2 + y$	$z = 5$
H_2			**G**

Figure 6: A Vector Relation Table

3.7 Mixed Vector Tables

A *mixed vector table,* T, is a table in which the elements of the main grid, G, are either predicate expressions or terms, the elements of H_1, H_3, ..., $H_{dim(T)}$ are predicate expressions, and the elements of H_2 are single variables. For any particular 2-stripe of G either all the elements are terms or all the elements are predicate expressions, i.e. no 2-stripe of G contains a mixture of predicate expressions and terms. T will be interpreted as a relation whose domain is a set of assignments and range is a set of $len_1(T)$-tuples. The variables that are the elements of H_2 are used to associate individual elements of the $len_1(T)$-tuples in the range with elements of the assignments in the domain of the relation.

A mixed vector table, T, is said to be *proper* if and only if:

- for any indices I, J of T, the conjunction of $T_{I,1}$, $T_{I,3}$, $T_{I,4}$, ..., $T_{I,dim(T)}$, $T_{J,1}$, $T_{J,3}$, $T_{J,4}$, ..., $T_{J,dim(T)}$ is equivalent to **false** when $I \neq J$.

[7] It is a consequence of this definition that the values corresponding to the variables mentioned in H_2 in the assignment a can affect the domains of the sub-relations $R_{I|1}$ but cannot restrict the values in the range. It is often best not to use any variable that is used in G in H_2.

- In H_2, all stripes other than 1-stripes are constant,
- No two elements of a 1-stripe of H_2 are the same variable,

Let I be in the index set of T. A relation $R_{I|1}$ can be associated with I|1 as follows

- Let K_I be the conjunction of $T_{I,1}$, $T_{I,3}$, $T_{I,4}$, ..., $T_{I,dim(T)}$.
- Let V be a tuple with length = $len_1(G)$. V_j will be associated with the variable that is j^{th} element of the 1-stripes of H_2.

A pair (a, V) is in $R_{I|1}$, if and only if, a is in the set of assignments characterised by K_I and, for all positive integers $j \leq len_1(G)$:

- if $G^2_{(I|1|j)|2}$ contains predicate expressions and $T_{I,2}$ is x_k, $a[k \rightarrow V_j]$ [8] satisfies $G_{I|1|j}$,
- if $G^2_{(I|1|j)|2}$ contains terms, V_j = the value of $G_{I|1|j}$ for assignment a..

A proper mixed vector table, T, can be interpreted as a relation, R_T, that is the union of the relations $R_{I|1}$ for all I in the index set of T.

Note that even if all the 2-stripes contain predicate expressions, a mixed vector table is not necessarily equivalent to an identical vector relation table. In the latter, the whole vector must satisfy all the predicates in the 2-stripe; in mixed vector tables each element is considered separately and must only satisfy the predicate in its 2-stripe.

When these tables are printed is helpful to print "=" in the headers of the rows containing terms and "|" in the headers of the rows containing predicate expressions.

	$w < o$	$w = o$	$w > o$
			H_1
$x =$	$x + w + q$	$x + 2 - q$	$x - w$
$y \mid$	$y^2 = x + 2$	$y = x + 2$	$y = \lvert x \rvert + 2$
$z \mid$	$z^2 = x^2 + y^2 + w^2$	$z^2 = x^2 + w^2$	$z = 5$
H_2			**G**

Figure 7: A Mixed Vector Table

3.8 Predicate Expression Tables

A *predicate expression table*, T, is a table in which the elements of the main grid, G, and all headers are predicate expressions. T will be interpreted as a predicate whose domain is a set of assignments.

A predicate expression table, T, is said to be *proper* if and only if, for any indices I, J of T, the conjunction of $T_{I,1}$, $T_{I,2}$, ..., $T_{I,dim(T)}$, $T_{J,1}$, $T_{J,2}$, ..., $T_{J,dim(T)}$ is equivalent to **false** when $I \neq J$.

[8] As in [9], A[k⟶c] stands for the assignment obtained from A by replacing the k^{th} element by c.

An index of T, I, will be associated with a predicate expression P_I that is the conjunction of $T_{I,1, ..., } T_{I,dim(T)}$ and G_I.

A proper predicate expression table, T, is equivalent to the disjunction of the expressions P_I for all I in the index set of T.

.

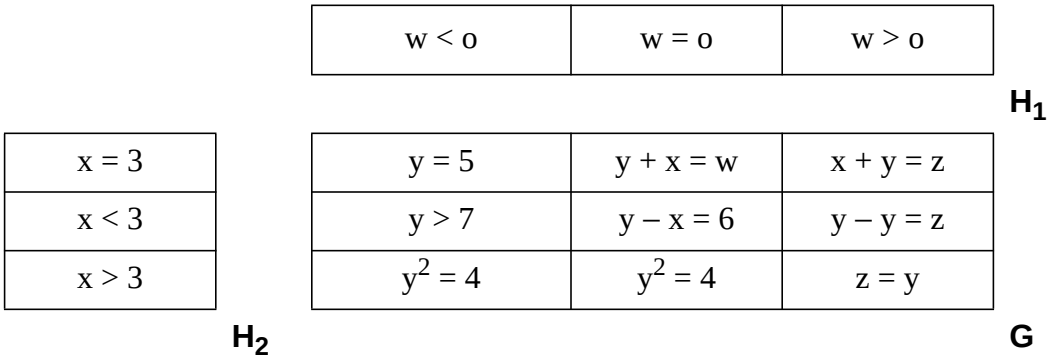


Figure 8: A Predicate Expression Table

3.9 Characteristic Predicate Tables

A *characteristic predicate table*, T, is a table in which the elements of the main grid, G, and all headers are predicate expressions. T will be interpreted as a relation whose range and domain comprise tuples of a fixed length, n. Each element of the tuple will be identified by a variable.In this definition, we shall assume that the variables are x_1, x_2, ..., x_n and that x_i is associated with the i^{th} position in the tuples. The variables x_1, x_2, ..., x_n themselves do not appear in these tables; instead we find a subset of the variables ‘x_1, ‘x_2, ..., ‘x_n, x’$_1$, x’$_2$, ..., x’$_n$. The *decorations* (the symbols “‘” and “’”) are considered part of the variable name.

A characteristic predicate table, T, is said to be *proper* if and only if, for any indices I, J of T, the conjunction of $T_{I,1}$, $T_{I,2}$, ..., $T_{I,dim(T)}$, $T_{J,1}$, $T_{J,2}$, ..., $T_{J,dim(T)}$ is equivalent to **false** when $I \neq J$.

An index of T, I, will be associated with a predicate expression P_I that is the conjunction of $T_{I,1}$, ..., $T_{I,dim(T)}$ and G_I.

A proper characteristic predicate table, T, is equivalent to a conventional boolean expression that is the disjunction of the expressions P_I for all I in the index set of T. This boolean expression is interpreted as the characteristic predicate of a set of ordered pairs of the form:

$$((‘x_1, ‘x_2, ..., ‘x_n), (x’_1, x’_2, ..., x’_n)).$$

This set of ordered pairs is a relation that is the intended interpretation of T.

Strictly, this form of table cannot be interpreted unless a list of variables is supplied. Without such a list, the length of the tuples and the association between variables and tuple positions is unknown. If no list is supplied, it is assumed that the only variables are the undecorated versions of the variables that appear in the table and that the order of values in the tuples can be arbitrarily chosen.

We have used the convention that the appearance of an undecorated variable, x_i, in G_I is

equivalent to conjoining "'$x_i = x'_i$" to P_I. It should be noted that if there is no occurrence of x_i, 'x_i, or x'_i in a table, the above interpretation includes all values in the Universe in the i^{th} position of the elements of the ordered pairs in the set. Thus, omissions of this sort amount to a "don't care" assertion.

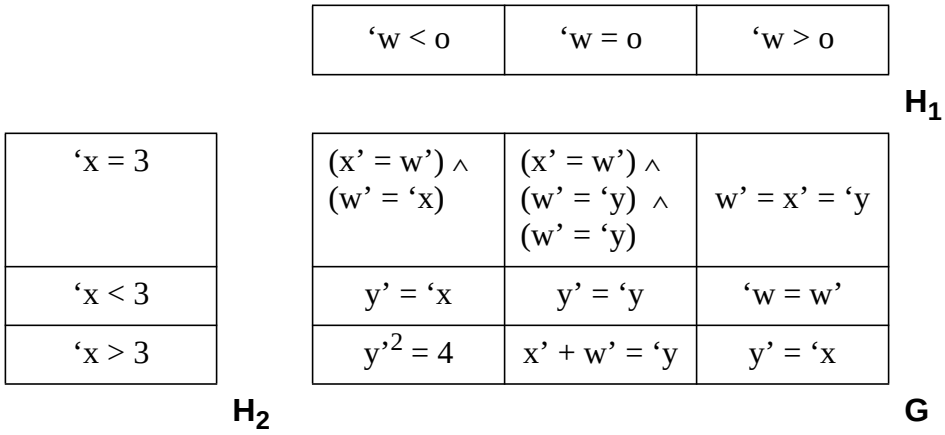


Figure 9: A Characteristic Predicate Table

3.10 Generalised Decision Tables

A *generalised decision table, T,* is a table in which the elements of the main grid, G, are predicate expressions that may contain a distinguished variable, which we shall denote by "#", and the elements of H_1 and H_2 are terms that do not include "#". H_3, ..., $H_{dim(T)}$ are not used in this interpretation of tables.

Let a be an assignment and a^i be the assignment obtained from a by replacing the element of a that corresponds to # with the value of $T_{I|2|i,1}$ for assignment a. $G^2_{I|2}$ is said to *apply* for a if, for all i, $1 \leq i \leq len_2(G)$, the predicate expression $G^2_{I|2,i}$ evaluates to **true** for the assignment a^i.

A generalised decision table, T, is said to be *proper* if and only if there is no assignment for which two distinct 2-stripes apply.

Let I be in the index set of T. I|2 can be associated with a function $F_{I|2}$ as follows:

- The domain of $F_{I|2}$ is the set of assignments for which $G^2_{I|2}$ applies.
- Within that domain, the value of $F_{I|2}$ is determined by evaluating $T_{I,2}$.

A proper generalised decision table, T, can be interpreted as a function, F_T, that is the union of the functions $F_{I|2}$ for all I in the Index set of T.

It is often useful to have the elements of H_2 be 1-dimensional grids; the values of the function F_T are then tuples. A 2-dimensional special class of these tables, in which the elements of H_1 are integer valued expressions and the elements of the main grid are inequalities, has been thoroughly studied and found useful in many applications [11].

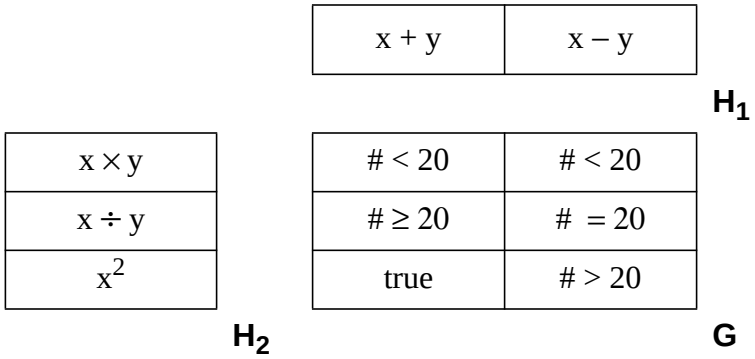


Figure 10: A Generalized Decision Table

4 Strictly Proper

In each of our definitions, if a table is proper, one can be certain that the domains of the subfunctions are disjoint. If a reader has found an entry that applies to the case at hand, no other entries need be considered. However, a stronger condition, which we will call “strictly proper”, makes the tables easier to use.

An integer, i, is called a *coordinate dimension* of a table, T,

- for normal function tables, normal relation tables, predicate expression tables, and characteristic predicate tables if $1 \leq i \leq \dim(T)$,
- for inverted function tables and inverted relation tables if $2 \leq i \leq \dim(T)$,
- for vector relation tables, vector function tables, and mixed vector tables if $1 \leq i \leq \dim(T)$ and $i \neq 2$.

A table, T, of one of the above types is *strictly proper* if, for any coordinate dimension, i, all the i-headerstripes are disjoint and all are identical.

In strictly proper tables, one need give only one headerstripe for each coordinate dimension instead of the full headers. To evaluate a function or relation one can determine the relevant coordinates of the table element by looking at the headers in any order.

5 Abbreviated Grids

Experience has shown that grids used as headers often contain subexpressions that are repeated in many elements. We have developed four types of grids that can be used to provide abbreviations for the grids described above. We will define each of these by describing its equivalent standard (non-abbreviated) grid. The components of abbreviated grids are assumed to be standard grids, but these may be expansions of abbreviated grids.

5.1 Conjunction Grids

A *conjunction grid*, G, is a grid with shape = <2> in which $G_{<1>}$ is a predicate expression and

$G_{<2>}$ is a standard grid whose elements are predicate expressions. The standard grid equivalent to G has the same shape as $G_{<2>}$. The elements of the equivalent grid are the conjunction of $G_{<1>}$ (parenthesised) with the (parenthesised) corresponding elements of $G_{<2>}$.

$x^2 + y^2 > 4$		
$x + y < 4$	$x + y = 4$	$x + y > 4$

This is an abbreviation for:

$(x^2 + y^2 > 4) \wedge (x + y < 4)$	$(x^2 + y^2 > 4) \wedge (x + y = 4)$	$(x^2 + y^2 > 4) \wedge (x + y > 4)$

Figure 11: A Conjunction Grid

5.2 Union Grids

A *union grid*, G, is a grid with shape = <2> in which $G_{<1>}$ is a predicate expression and $G_{<2>}$ is a standard grid whose elements are predicate expressions. The elements of the equivalent grid are the union of $G_{<1>}$ (parenthesised) with the (parenthesised) corresponding elements of $G_{<2>}$.

$x^2 + y^2 > 4$		
$x + y < 4$	$x + y = 4$	$x + y > 4$

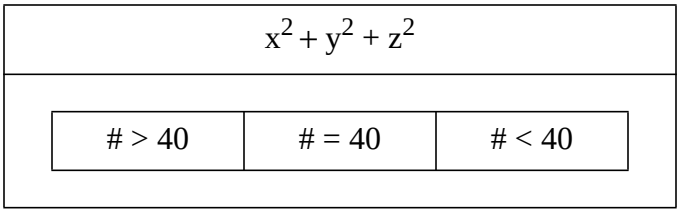
This is an abbreviation for:

$(x^2 + y^2 > 4) \vee (x + y < 4)$	$(x^2 + y^2 > 4) \vee (x + y = 4)$	$(x^2 + y^2 > 4) \vee (x + y > 4)$c

Figure 12: A Union Grid

5.3 Substitution Grids

A *substitution grid*, G, is a grid with shape = <2> in which $G_{<1>}$ is a predicate expression or a term and $G_{<2>}$ is a standard grid in which a distinguished variable, written "#", may appear. The equivalent standard grid is obtained from $G_{<2>}$ by replacing all occurrences of "#" with a parenthesised copy of $G_{<1>}$.



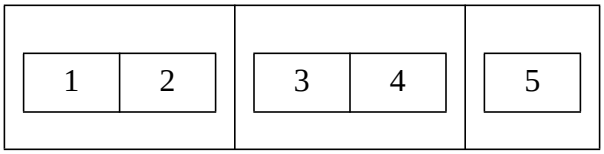
This is an abbreviation for:

$(x^2 + y^2 + z^2) > 40$	$(x^2 + y^2 + z^2) = 40$	$(x^2 + y^2 + z^2) < 40$

Figure 13: A Substitution Grid

5.4 Concatenation Grids

A *concatenation grid* is a 1-dimensional grid whose elements are standard 1-dimensional grids. The equivalent standard grid is a 1-dimensional grid whose length is the sum of the lengths of the elements of the concatenation grid and whose elements are those of the component grids in the same order as they appeared in the concatenation grid.



This is an abbreviation for:

1	2	3	4	5

Figure 14: A Concatenation Grid

6 Using Tables for Recursive Definitions

Tabular expressions can be used in recursive definitions of functions. The function being defined may be applied in the table defining it. Each such table will be equivalent to a more conventional mathematical expression; the rules governing recursive definitions are unchanged.

7 Relations and Functions viewed as Predicates

Tables interpreted as relations may also be interpreted as the characteristic predicate of the set of ordered pairs that constitutes the relation. Consequently, such tables may be considered to be predicate expressions and be part of larger predicate expressions. Those larger expressions can be interpreted as relations since they characterise a set of ordered pairs. Note that if a table happens to denote a function that is a predicate, this interpretation yields a different predicate. Such tables must be tagged to indicate which interpretation is desired.

8 Conclusions and Future Work

Our work in both defining and using these tables makes it clear that this paper is "just the beginning". We expect that many new and useful table forms can be found and defined. Moreover, the formal definition of these tables has led us to more general and powerful notations, notations that allow even more compact description of functions. We expect that future efforts to improve the definitions of the various table forms will lead to improvements in both the definitions and the notations that are defined.

9 Acknowledgements

Kathryn Britton and John Shore played important roles in the early use of tables in the NRL "A-7" project. Rod Byrne and Daniel Hoffman provided a helpful response to my first draft of this paper, demonstrating the advantages of treating the headers as separate grids. Z. Pawlak inspired the idea of generalised decision tables with a lecture on conventional decision tables. Michal Iglewski and Jan Madey participated in many discussions about table formats. Ramesh Bharadwaj and Michal Iglewski provided helpful suggestions on recent drafts of this paper. Ryszard Janicki provided thoughtful comments on the definition of the semantics of the table forms. Dennis Peters helped to improve the definition of "proper" for tables with dimensionality greater than 2.

10 References

1. Parnas, D.L., Madey, J., "Functional Documentation for Computer Systems Engineering", *Technical Report 90-287*, Queen's University, C&IS, Telecommunications Research Institute of Ontario (TRIO), Kingston, Ontario, Canada, September 1990, 14 pp.
2. Parnas, D.L., Madey J., "Functional Documentation for Computer Systems Engineering (Version 2)", *CRL Report 237*, McMaster University, Hamilton Canada, TRIO (Telecommunications Research Institute of Ontario), September 1991, 14 pgs.
3. Parnas, D.L., Asmis, G.J.K., Madey, J., "Assessment of Safety-Critical Software in Nuclear Power Plants", *Nuclear Safety*, vol. 32, no. 2, April-June 1991, pp. 189-198.
4. Heninger, K.L., "Specifying Software Requirements for Complex Systems: New Techniques and their Application", *IEEE Transactions Software Engineering*, Vol. SE-6, No. 1, January 1980, pp. 2-13.
5. Heninger, K.L., Kallander, J., Parnas, D.L., Shore, J.E., "Software Requirements for the A-7E Aircraft", *NRL Memorandum Report 3876*, United States Naval Research Laboratory, Washington D.C., November 1978, 523 pp.
6. Hester, S.D., Parnas, D.L., Utter, D.F., "Using Documentation as a Software Design Medium", *Bell System Technical Journal*, Vol. 60, No. 8, October 1981, pp. 1941-1977.
7. van Schouwen, A.J., "The A-7 Requirements Model: Re-examination for Real-Time Systems and an Application to Monitoring Systems", *Technical Report 90-276*, Queen's University, C&IS, Telecommunications Research Institute of Ontario (TRIO), Kingston, Ontario, Canada, May 1990, 93 pp. Reprinted as CRL Report 242 (Communication Research Laboratory, McMaster University).
8. Clements, P.C., "Function Specifications for the A-7E Function Driver Module", *NRL*

Memorandum Report 4658, 27 November 1981.

9. Parnas, D.L., “Predicate Logic for Software Engineering”, CRL Report 241, McMaster University, Hamilton, Canada, TRIO (Telecommunications Research Institute of Ontario), February 1992, 8 pgs.
10. Mills, H.D., Basili, V.R., Gannon, J.D., Hamlet, R.G., *Principles of Computer Programming: A Mathematical Approach*, Allyn and Bacon, 1987.
11. Hurley, R.B., *Decision tables in software engineering*, Van Nostrand Reinhold Company, 1983.
12. Halmos, P. R., “*Naive Set Theory*”, Van Nostrand Reinhold Company, 1960.

11:05 a.m.
10-14-92