Description of a set-theoretic data structure

by DAVID L. CHILDS

University of Michigan
Ann Arbor, Michigan

INTRODUCTION

The overall goal, of which this paper is a part, is the development of a machine-independent data structure allowing rapid processing of data related by arbitrary assignment such as: the contents of a telephone book, library files, census reports, family lineage, graphic displays, information retrieval systems, networks, etc. Data which are non-intrinsically related have to be expressed (stored) in such a way as to define the way in which they are related before *any* data structure is applicable. Since any relation can be expressed in set theory as a set of ordered pairs and since set theory provides a wealth of operations for dealing with relations, a set-theoretic data structure appears worth investigation.

A Set-Theoretic Data Structure (STDS) is a storage representation of sets and set operations such that: given any family of sets η and any collection S of set operations an STDS is any storage representation which is isomorphic to η with S. The language used with an STDS may contain any set-theoretic expression capable of construction from η and S. Every stored representation of a set must preserve all the properties of that set and every representation of a particular set must behave identically under set operations.

General storage representation

An STDS is comprised of five structurally independent parts:

1) a collection of set operations S.
2) a set of datum names β.
3) the data: a collection of datum definitions, one for each datum name.
4) a collection of set names η.
5) a collection of set representations, each with a name in η.

The storage representation is shown schematically in Figure 1. In order for an STDS to be practical the set operations must be executed rapidly. If any two sets can be well ordered (a linear order with a first element) such that their union preserves this well-ordering, then the subroutines needed for set operations just involve a form of merge or, at worst, a binary search of just one of the sets. It was shown in another paper[1] that *any* set defined over β could be so ordered. Sets are represented by blocks of contiguous storage locations with η containing names of all the sets. The set β is the set of all datum names, and is represented by a contiguous block of storage locations; the address of a location in the β-block is a datum name and an element of β. The content of a location in the β-block is the address of a stored description of that datum (see Figure 1). The contents of the β-block and the η-block are the *only* pointers needed for the operation of an STDS. The storage representations of the individual sets *do not* contain pointers to other sets, but contain information about datum names. Since each set representation has only one pointer associated with it, the set representation can be moved throughout storage without affecting its contents or the contents of any other set representation—only the one pointer in η is affected. Updating set representations is virtually trivial. Elements to be deleted are replaced by the last element in the set. Elements to be added are added to the end of the set representation as space allows. When contiguous locations are no longer available a new set is formed and the element in η that referenced the set before it was extended now references a location that indicates that the set is now the union of two set representations. (In a paging structure such sets could be kept on the same page.) This demonstrates two different kinds of sets in η: *generator* sets and *composite* sets. Only the generator sets have storage representations, the composite sets are unions of generator sets, and the generator sets are mutually disjoint. Since no duplication of storage of sets is necessary and since the set representa-

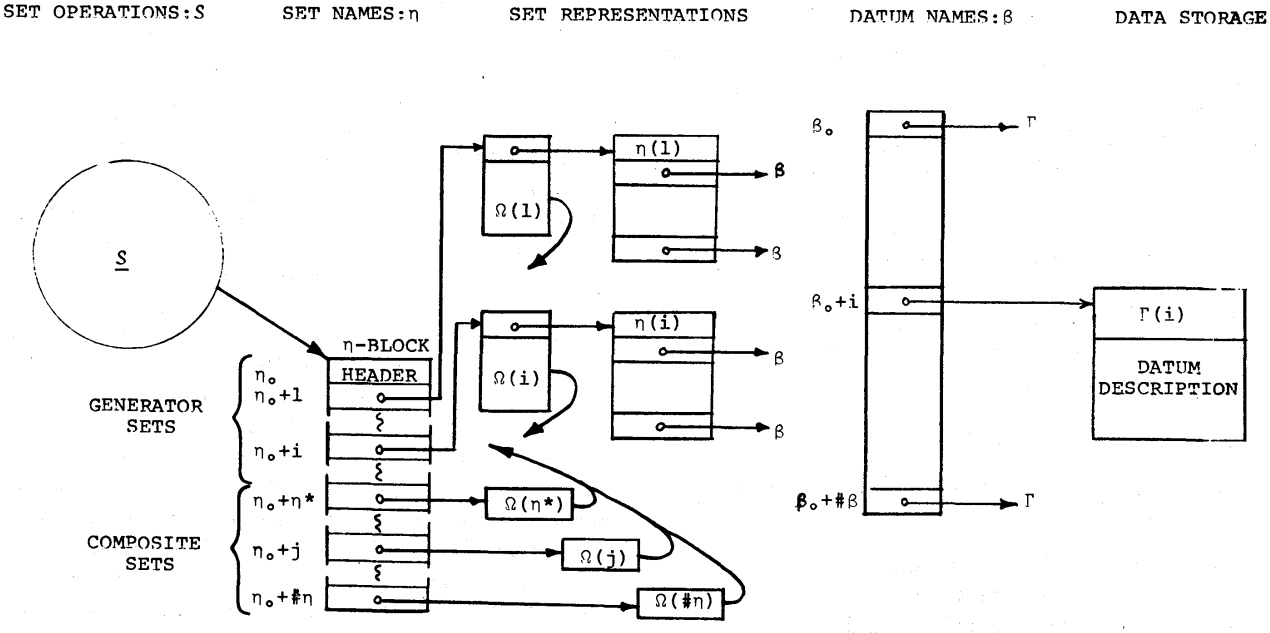
tions are kept to a minimum by containing just the elements of the sets and no pointers, an STDS is intrinsically a mimimal storage representation for arbitrarily related data.

Operation of an STDS

An STDS relies on set operations to do the work usually allocated to pointers or hash-coding as in list structures, ring structures, associative structures, and relational files. A set operation of S is represented by a subroutine which accesses sets through pointers in η. Again it should be stressed that no pointers exist between sets, hence the set operations S act as the *only* structural ties between sets. Since S will allow any set-theoretic operation, S will be rich enough that all information between sets may be expressed by a set-theoretic expression generated from the operations of S. Any expression establishes which sets are to be accessed and which operations are to be performed within and between these sets; therefore all pages needed for completion of an expression are known before the expression is executed. Complementing the set operation subroutines are some strictly storage manipulation subroutines. These, however, are not reflected in any set-theoretic expression. These routines change storage modes and perform sorts and orderings. A fast sort routine has been programmed with execution times as a linear function of the number of words to be sorted. (On an IBM 7090 this sort ordered 1000 words in 0.35 seconds and 10,000 words in 3.3 seconds. The nature of this sort is such that on an IBM 360/67 it may sort up to 60,000 bytes per second. This routine is presently being programmed.) Another subroutine which is crucial to the operation of an STDS is the tau-ordering routine.[1] This routine gives a well-ordering which is preserved under union.

Details of β-block

The β-block is a section of contiguous storage locations with β_0 as the address of the head location. The first location containing a datum-pointer has the address β_0+1, and the location of the i-th datum-pointer is β_0+i. Let $\#\beta$ represent the total number of datum-pointers, then the last address of the β-block would be $\beta_0+\#\beta$. β is the set of datum-names or locations of datum-pointers in the β-block. Since all datum-pointers are located between β_0+1 and $\beta_0+\#\beta$, let β be the set of integers $\{1,2.\ .\ .,\#\beta\}$. Therefore any integer i such



Ω(1) through Ω(η*-1) are sets of pointers to COMPOSITE SETS in η-BLOCK

Ω(η*) through Ω(#η) are sets of pointers to GENERATOR SETS in η-BLOCK

Γ(i) are sets of pointers to GENERATOR SETS in η-BLOCK

FIGURE 1

that $1 \leq i \leq \#\beta$ is the datum-name for the i-th datum-pointer. The i-th datum-pointer locates a block of storage containing a description of the i-th datum and all the generator set names (elements of η) for which the i-th datum name is a constituent.

Details of η-block

The η-block is similar to the β-block with η_0 and $\#\eta$ as the address of the head location and cardinality respectively. The contents of the η-block are pointers. These pointers are of two types and are distinguished by an integer η^* such that $1 < \eta^* \leq \#\eta$. For all $1 \leq i < \eta^*$, i is the name of a generator set, and for all $\eta^* \leq i \leq \#\eta$, i is a composite set. A generator set has a set representation while a composite set does not since it is the union of some generator sets. For $i \geq \eta^*$ the pointer in $\eta_0 + i$ locates a section of storage containing names of generator sets. For $i < \eta^*$ the pointer in $\eta_0 + i$ locates a section of storage containing all composite set names that use i, and a pointer to the set representation of i. Since all generator sets are mutually disjoint and since only generator sets have a storage representation, there is no duplication of storage in an STDS.

Set representation

In order to insure fast execution times for the set operations in S, the sets involved must be isomorphic to a *unique* linear representation of their elements. Unique is used here to mean unique relative to some predefined well-ordering relation, such that independently of how the set is presented to a machine the ordering of its elements will always be the same. This well-ordering must be preserved under union. Any ordering satisfying the above conditions is adequate for the efficient operation of an STDS.[1]

Since the set representatives must be isomorphic to the sets they represent, every set representation must reflect the rank and preserve the order (if any) of the sets and their elements. Let A = <a,b,c>, B = {a, b,c}, and C = {c,b,a} then B and C must have the same set representation while A must have a completely different representation. For simple sets like these, adequate representations are trivial, such is not always the case, however.

Complexes and n-tuples

If an STDS is to be general then it will have to accommodate more imaginative sets than the ones above. Let W = {a,b,{{c}},<a,{b,d},c>,<<a,b>, c>} and V = {<a,b,c>,<{<a,b>,<c,d>},<d,a, >>,{{c}},b}. In order for set operations on these sets to fall within the allotted time bounds, the storage representations of W and V must satisfy the well-ordering condition. Such a representation is not immediately obvious. Two problems arise. (1) The first problem is machine oriented in that an ordered set in set theory is defined through nesting and repetition of the elements of the set. For example the Kuratowski definition of ordered pair gives <a,b> = {{a},{a,b}}. Since any machine representation will induce an order on the elements of a set by their location in storage, this may be utilized instead of relying on redundancy of storage. This in turn may present problems in preserving the isomorphism between sets and their set representations, since an unordered set must have a unique representation and no ordering on its elements. (2) The second problem is much allied with the first except that it is more biased towards the foundations of set theory. There seems to be a general lack of precision in set theory when ordering beyond a pair is involved. No set representation of ordered triples, ordered quadruples, quintuples, sextuples, etc., is given save for an arbitrary assignment in terms of ordered pairs. (This problem is discussed by Skolem[3]). For example <a,b,c,d> has no set equivalent independent of ordered pairs, it is given one of the following as its canonical form: <<a,b>, <c,d>>; <a,<b,<c,d >>>; <a,<<b,c>,d>; <<<a,b>,c>,d>;<< a,<b,c>>,d>; or {<1,a>, <2,b>, <3,c>, <4, d>}. Clearly each of these sets has independent stature and assigning one as a canonical form of the other precludes the use of the others. The problem with ordered tuples is compounded in that though they are defined as sets they are excluded from meaningful set operations. The intersection between quadruples <a, b,c,d> and <x,b,c,d> is always empty unless a = x, and even then it depends on which assignment is used. In another paper[1] the definition of a "complex" is presented which preserves the distinction between different nestings of ordered pairs, does not require order to be defined by repetition, and does not arbitrarily exclude certain sets from being operated on by set operations. The formal definition of a complex is given by the following, where N is the set of natural numbers.

DEFINITION OF A COMPLEX: Any two sets A and B form a complex (A;B) if and only if $(\exists X)$

$(\exists Y)(X \in \{A,B\})(Y \in \{A,B\})[(\forall x \in X)(\exists i \in N)$

$(\{\{x\},i\} \in Y) \,\&\, (\forall y \in Y)(\exists j \in N)(\exists x \in X)(\{\{x\},j\} = y)]$

This definition is stated in such a way as not to presuppose any ordering in (A;B) of A before B, insuring that a complex be an unordered coupling of two sets, each bearing a mutual dependence on the other. The defi-

nition states that for every element x of one of the sets, X, the other set, Y, contains an element containing a natural number and a set whose only element is x; and that Y is such that every element of Y contains only a natural number and a singleton set containing an element of X (either X=A and Y=B, or X=B and Y=A, but not both). Let A={a,b,c}, B={{{a},1}, {{b},3},{{c}, 963},{{b},6}} and let C={a,b,{{b},3}, {{a},1},{{d},6}} then (A;B), (B;A) and (A∩C;B∩C) are complexes, while (A;A), (A;C), (A;B∩C) and (A∩C;B) are not complexes. From the definition it should be noticed that if (A;B) is a complex then (B;A) is the same complex and A≠B. Without giving a formal definition here let $x \epsilon_i A$ be understood to mean that x is in the i-th position of the complex A, then a notational schema for a complex is given by:

DEFINITION SCHEMA: $\{x^i : \Psi(x,i)\} = A$ iff $[(\forall x)$ $(\forall i \epsilon N)\ (x \epsilon_i A \leftrightarrow \Psi(x,i))$ & A is a complex]

These results allow a set theoretic foundation for the following equivalent notations:

set	$\{a,b,c\} = \{a^1,b^1,c^1\}$
ordered pair	$<a,b> = \{a^1,b^2\}$
ordered triple	$<a,b,c> = \{a^1,b^2,c^3\}$
ordered quadruple	$<a,b,c,d> = \{a^1,b^2,c^3,d^4\}$
ordered pairs of ordered pairs	
	$<<a,b>,<c,d>> = \{\{a^1,b^2\}^1,\{c^1,d^2\}^2\}$
	$<a,<b,<c,d>>> = \{a^1,\{b^1,\{c^1,d^2\}^2\}^2\}$
	$<a,<<b,c>,d>> = \{a^1,\{\{b^1,c^2\}^1,d^2\}^2\}$
	$<<<a,b>,c>,d> = \{\{\{a^1,b^2\}^1,c^2\}^1,d^2\}$
	$<<a,<b,c>>,d> = \{\{a^1,\{b^1,c^2\}^2\}^1,d^2\}$
$\{<1,a>,<2,b>,<3,c>,<4,d>\}$	$= \{\{1^1,a^2,\},\{2^1,b^2\}, \{3^1,c^2\};\{4^1,d^2\}\}$

and from the beginning of this section,

$$W = \{a^1,b^1,\{\{c^1\}\},\{a^1,\{b^1,d^1\}^2,c^3\},\{\{a^1,b^2\},c^1\}\}$$

$$V = \{\{a^1,b^2,c^3\},\{\{\{a^1,b^2\},\{c^1,d^2\}\},\{d^1,a^2\}^2\},\{\{c^1\}\},b^1\}$$

Since for all a, $\{a^1\} = \{a\}$, the exponent '1' is optional. It should be stressed that the symbol 'x^i' has no meaning apart from being enclosed by set brackets. mean- $\{a^6,b^8\}$, then $a \epsilon_6 A$ and $b \epsilon_8 A$ are true, but $a^6 \epsilon A$ isIf A= ingless. For examples of set operations between complexes see Figure 2.

1) <a,b,c>∩<x,b,y> = {b²}

2) <a,b,c>∪<x,y> = {a¹,x¹,b²,y²,c³}

3) {a,b,c}∩<a,x,y> = <a> = {a¹} = {a}

4) ∪{a¹,b²,{x¹,c³}³,{y²,d⁴}⁴} = {x¹,c³}∪{y²,d⁴} = <x,y,c,d>

5) <a,b,z>△<a,y,c>△<x,b,c> = <x,y,z>

6) <a,b,c,d> ∿ <x,y,c,d> = <a,b>

FIGURE 2—Set operations between complexes

Set operation subroutines

The viability of an STDS rests not only on the speed of the set operations, but also on their scope. Table 1 presents some available set operations for constructing questions in any way compatible within a parent language. (For those who are not familiar with the set-theoretic definitions or are not accustomed to the notation preferred in this monograph, the definitions are given in Appendix I.) These subroutines are presented in a format compatible with FORTRAN, and with MAD if periods are added as in the examples to follow. The argument represented by C in the subroutines can be deleted. This default case assigns a temporary storage block whose location is returned in D, as if it were a permanent storage location, i.e., D = UN(A,B). Since all subroutines operate on the name of a storage block representing a set, then for all subroutines that return a name, any degree of nesting of these subroutines within subroutines is allowable (see examples). Since the only restriction on a set representation is that it be isomorphic to the set and have a predefined well-ordering on its elements, there are many storage configurations available. MODE allows a choice of different storage configurations for non-set-theoretic needs. Though all the subroutines appear to be defined just for sets, they are defined for any complex as well. However, to make use of complexes that are not sets since they allow the extension of binary relation properties (e.g., domain, image, relative product, restriction, etc.) to sets of arbitrary length n-tuples, further delimiters must be included. For example using 'Q' and an extra argument the I-th relative produce of A with B could be QRP (I, A, B, C), and the I-th domain of A could be QDM(I, A, C), and QELM(I, A, B) could represent the question "is A an I-th element of B?"

Some applications

This section will be devoted to examples demonstrating the applicability of set-theoretic questions. For a germane reference on computer graphics see Johnson.[2] The first two examples are to give some indication of execution times. The two examples were run on an IBM 7090, the times may or may not be characteristic of the potential speeds in an STDS. With just two examples no claims can be made other than that two examples were run with the following results:

EXAMPLE 1: Given a population of 24,000 people and a file F containing a ten-tuple for each person such that each ten-tuple is of the form <age, sex, marital status, race, political affiliation, mother tongue, employment status, family size, highest school grade completed, type of dwelling>, the following four questions were asked:

a. Find the number of married females:
   Answer: 6,015 Time: 0.50 seconds
b. Find the number of people of Spanish race whose mother tongue is not Spanish.
   Answer: 1,352 Time: 0.48 seconds
c. Find the number of people aged 93 or 94.
   Answer: 46 Time: 0.73 seconds
d. Find the number of males and unmarried females.
   Answer: 17,985 Time: 0.55 seconds
e. Find the number of males between the ages of 20 and 40.
   Answer: 588 Time: 0.62 seconds.

EXAMPLE 2: Given a population of 3,000 people and given two collections, A and B, of subsets from this population such that: A contains 20 sets of 500 people, and B contains 500 sets of 20 people. Find the set of people belonging to *some* set in A, to *all* sets in A, and to an *odd* number of sets in A; and similarly for B.

Results	*A-Times*	*B-Times*
a. people in some set	0.73 sec	0.76 sec
b. people in all sets	0.48 sec	0.05 sec
c. people in odd no. of sets	0.76 sec	0.78 sec

A point to notice is that where every element has to be accessed, as in (a) and (c), the times are dependent on the total number of elements included ($\xi(A) = \xi(B) = 10{,}000$) and not the number of sets involved (20 for A and 500 for B).

Examples three and four are presented with MAD as the parent language, therefore all the subroutine names must end with a period.

EXAMPLE 3: Let six sets A,B,C,D,E, and F be the membership lists of six country clubs. For each male resident of Ann Arbor, let there be a datum in β for a data-block containing: person's name, address, phone number, credit rating, age, golf handicap, wife's name (if any), political affiliation, religious preference, and salary. The set η will contain the names of the sets, namely: A(0), B(0), C(0), D(0), E(0), F(0). This along with the collection $\mathcal{S}$ of set operations allows answering the following questions.

1) How many members belong to club A or B but not C?
2) Find the phone numbers of members in an odd number of clubs.
3) Get addresses of members belonging to one and only one club.
4) Get addresses and phone numbers of people not in any club.
5) Find members of A that are not also in B but who may be in C only if they are not in D, or in E if they are not in F.
6) Get the average credit rating of members belonging to exactly three clubs.

The possible questions may become ridiculously involved and may interact with any spontaneously constructed sets. For example of the latter, let X be the set of Ann Arbor males born in Ann Arbor.

7) Find the average age of members born in Ann Arbor and compare with average age of members not born in Ann Arbor.

The answers to (1) through (7) formulated in an STDS are expressed below, with N and M representing real numbers, and with BB for β and NN for η.

1) N = C. (RL. (UN. (A,B),C))
   ans: N
2) ACC. (1,SD. (1,NN),Q)
   ans: Q Format 1 gives phone numbers (see Table 1, #25)
3) ACC. (2,EX. (1,NN),Q)
   ans: Q Format 2 gives addresses
4) ACC. (3,RL. (BB,UN.(1,NN)),Q)
   ans: Q Format 3 gives phone numbers and addresses
5) RL.(RL.(A,B),UN.(RL.(D,C),RL.(F,E)),Q)
   ans: Q
6) ACC.(4,EX.(3,NN),Q)
   N = 0
   THROUGH LOOP,
   FOR I=1,1,I.G.C.(Q)

```
LOOP  N = N + Q(I)
      N = N/C.(Q)
ans: N    Format 4 gives credit rating
```

```
7)        N = 0
          M = 0
          ACC.(5,X,T)
          THROUGH LOOP1,
          FOR I=1,1,I.G.C.(T)
LOOP1     N = N + T(I)
          ACC. (5,RL.(BB,X),P)
          THROUGH LOOP2,
          FOR I=1,1,I.G.C.(P)
LOOP2     M = M + P(I)
          N = N/C.(T)
          M = M/C.(P)
ans: N and M are the respective average ages
Format 5 gives ages
```

EXAMPLE 4: Family lineage is easily expressed in STDS. With just five initial relations defined over a population U, all questions concerning family ties may be expressed.

Let U be a population of people and let
$M = \{<x,y> : y \text{ is the mother of } x\}$
$F = \{<x,y> : y \text{ is the father of } x\}$
$S = \{<x,y> : y \text{ is a sister of } x\}$
$B = \{<x,y> : y \text{ is a brother of } x\}$
$H = \{<x,y> : y \text{ is a husband of } x\}$
Let X be any subset of the population U, find

1) the set G of grandfathers of X.

$G = F[(FUM)[X]]$	set notation
IM.(F,IM.(UN.(F,M),X),G)	in an STDS

2) the set GF of grandfathers of X on the father's side.

$GF = F[F[X]]$	set notation
IM.(F,IM.(F,X),GF)	STDS

3) the set GM of grandfathers of X on the mother's side

$GM = G \sim GF$	set notation
RL.(G,GF,GM)	STDS

4) the set GR: the grandfather relation over U.

$GR = (F \cup M)/F$	set notations
RP.(UN.(F,M),F,GR)	STDS

5) the general relation: $P = \{<x,y> : y \text{ is a parent of } x\}$

$P = F \cup M$	set notation
UN.(F,M,P)	STDS

6) the general relation: Sibling, L.

$L = S \cup B$	set notation
UN.(S,B,L)	STDS

7) the general relation: Children, C.

$C = \overline{M \cup F} = \overline{P}$	set notation
CV.(P,C)	STDS

8) the general relation: Aunt, A.

$A = (P/S) \cup (P/B/\overline{H})$	set notation
UN.(RP.(P,S),RP.(P,RP.(B,CV.(H))),A)	STDS

9) the general relation: Wife, W.

$W = \overline{H}$	set notation
CV.(H,W)	STDS

10) the general relation: Cousin, K.

$K = P/L/C$	set notation
RP.(P,RP.(L,C),K)	STDS

11) the general relation: Half-sibling, HS.

$HS = P/C \sim (M/\overline{M} \cup F/\overline{F})$	set notation
RL.(RP.(CV.(C),C),IN.(RP.(M,CV.(M)), RP.(F,CV.(F))),HS)	STDS

12) people in X with no brothers or sisters.

$Q = X \sim \mathfrak{D}(L)$	set notation
RL.(X,DM.(L),Q)	STDS

13) find all relations of X to a set Y such that Y is equal to the image of X.

$Q = \{A:(A \epsilon \eta)\ (Y = A[X])\}$	set notation
DC.(X,NN,T)	STDS

```
          THROUGH LOOP, FOR I=1,1,I.G.C.(T)
          B = IM.(T(I),X)
LOOP  WHENEVER EQL.(Y,B).E.1,
          UN.(Q,S.(T(I)),Q)
```

Many more possibilities are available and might be tried by the reader.

CONCLUSION

The purpose of an STDS is to provide a storage representation for arbitrarily related data allowing quick access, minimal storage, generality, and extreme flexibility. With the definition of a complex, a predefined well-ordering, and the operations of set theory, such a storage representation can be realized.

Set-theoretic definitions

Conventions

The logical connectives 'and,' 'or,' 'exclusive-or' are represented by '$\wedge$,' '$\vee$,' 'Δ.' 'For all x,' 'for some x,' 'for exactly n x' will be represented by '$\forall x$,' 'Ex', '$E(n)!x$.' Parentheses are used for separation, and as usual the concatenation of parentheses will represent conjunction.

'A' will be a *set* if and only if (a) it can be represented formally by abstraction (i.e., $A = \{x:\theta(x)\}$ where $\theta(x)$ is a predicate condition specifying the allowable elements 'x'); (b) 'A' can be represented by {,} enclosing the specific elements of 'A.'

Definitions

The symbol 'ϵ' means 'is an element of'; $x \epsilon A$ reads: "x is an element of A."

1) UNION
   a) binary union of two sets A and B
      $A \cup B = \{x:(x \epsilon A) \vee (x \epsilon B)\}$
   b) unary union of a family G of sets
      $\cup G = \{x:(\exists A \epsilon G)\ (x \epsilon A)\}$
   c) indexed union of a set f(A) over the family G
      $\cup_{A \epsilon G} f(A) = \{x:(\exists A \epsilon G)\ (x \epsilon f(A))\}$.
2) INTERSECTION
   a) binary intersection of A and B
      $A \cap B = \{x:(x \epsilon A)\ (x \epsilon B)\}$
   b) unary intersection of a family G
      $\cap G = \{x:(\forall A \epsilon G)\ (x \epsilon A)\}$
   c) indexed intersection of f(A) over the family G
      $\cap_{A \epsilon G} f(A) = \{x:(\forall A \epsilon G)\ (x \epsilon f(A))\}$.
3) SYMMETRIC DIFFERENCE
   a) binary symmetric difference of A and B
      $A \Delta B = \{x:(x \epsilon A) \Delta (x \epsilon B)\}$*
      * even though the symbol 'Δ' has two different meanings, no confusion is likely
   b) unary symmetric difference of G
      $\Delta G = \{x:$(for an odd number of $A \epsilon G$) $(x \epsilon A)\}$
   c) indexed symmetric difference of f(A) over G
      $\Delta_{A \epsilon G} f(A) = \{x:$(for odd no. of $A \epsilon G$) $(x \epsilon f(A))\}$.
4) RELATIVE COMPLEMENT
      $A \sim B = \{x:(x \epsilon A)(x \notin B)\}$.
5) EXACTLY N!
   the set of elements common to exactly 'n' elements of a given set G is represented by:
      $E_n G = \{x:(E(n)!A \epsilon G)(x \epsilon A)\}$.
6) DOMAIN of a set A
      $\mathfrak{D}(A) = \{x:(\exists y)(<x,y> \epsilon A)\}$*.
      * $<x,y>$ represents an ordered pair
7) RANGE of a set A
      $\mathfrak{R}(A) = \{y:(\exists x)(<x,y> \epsilon A)\}$.
8) IMAGE of B under A
      $A[B] = \{y:(\exists x \epsilon B)(<x,y> \epsilon A)\}$.
9) CONVERSE IMAGE of B under A
      $[B]A = \{x:(\exists y \epsilon B)(<x,y> \epsilon A)\}$.
10) CONVERSE of A
      $\overline{A} = \{<y,x>: <x,y> \epsilon A\}$.
11) RESTRICTION
      $A|B = \{<x,y>:(<x,y> \epsilon A)(x \epsilon B)\}$.
12) RELATIVE PRODUCT of A and B
      $A/B = \{<x,y>:(\exists z)(<x,z> \epsilon A)\ (<z,y> \epsilon B)\}$.
13) CARTESIAN PRODUCT of A and B
      $A \times B = \{<x,y>:(x \epsilon A)(y \epsilon B)\}$.
14) DOMAIN CONCURRENCE of X relative to A
      $\mathfrak{D}(X:A) = \{B:(B \epsilon A)(X \subset \mathfrak{D}(B))\}$.
15) RANGE CONCURRENCE of X relative to A
      $\mathfrak{a}(X:A) = \{B:(B \epsilon A)(X \subset \mathfrak{R}(B))\}$.
16) SET CONCURRENCE of X relative to A
      $\mathfrak{g}(X:A) = \{B:(B \epsilon A)(X \subset B)\}$.
17) CARDINALITY of A
      #A = n iff there are exactly n elements in A.
18) A is a *SUBSET* of B iff every element of A is an element of B:$CBB \leftrightarrow (\forall x)(x \epsilon A \rightarrow x \epsilon B)$.
19) A is *EQUAL* to B iff A is a subset of B, and B is a subset of A:$A = B \leftrightarrow (ACB \& BCA)$.
20) A and B are *DISJOINT* iff the intersection of A and B is empty:$A \cap B = \emptyset$.
21) A is *EQUIVALENT* to B iff A and B contain the same number of elements:$\#A = \#B$.

GLOSSARY OF SYMBOLS

Symbol	*Symbol Definition*
iff	if and only if
=	Identity
$\wedge$	Conjunction
v	Disjunction
Δ	Exclusive or
$\rightarrow$	Implication (if . . . then)
$\leftrightarrow$	Equivalence
$\forall x$	Universal quantifier (for all)
$\exists x$	Existential quantifies (for some)
$E!x$	Uniqueness quantifier (for exactly one)
Θx	Odd quantifier (for an odd number of)
$(En)!x$	Exact number quantifier
ϵ	Set membership
ϕ	Empty set
$\notin$	Non-membership
$\subset$	Set inclusion

$A \cap B$	Intersection	$A \times B$	Cartesian product
$A \cup B$	Union	$\mathfrak{D}(A)$	Domain of A
$A \Delta B$	Symmetric difference	$\mathfrak{R}(A)$	Range of A
$A \sim B$	Relative complement	$\overline{A}$	Converse of A
$<x,y>$	Ordered pair	A/B	Relative product of A and B
$\{x:\theta(x)\}$	Definition by abstraction	$A\|X$	A restricted to X
xAy	Ordered pair $<x,y>$ contained in A	$A[X]$	Image of X under A
$\cup G$	Union or sum of G	$[X]A$	Converse-image of X under A
$\cap G$	Intersection of G	$\mathfrak{D}(X)$	Domain-concurrence of X
ΔG	Symmetric difference of G	$\mathfrak{R}(X)$	Range-concurrence of X
$E_n G$	Elements contained in exactly n elements of G	$\mathfrak{S}(X)$	Set-concurrence of X
		$\xi(A)$	Total cardinality of A

The last column contains an executable expression of the set-theoretic expression preceding it. D is an indirect name for the permanent storage with name C, or for temporary storage if the argument C is deleted, (see text).

1) UNION	$C = A \cup B$	D = UN (A, B, C)
	$C = \cup A$	D = UN (1, A, C)
2) INTERSECTION	$C = A \cap B$	D = IN (A, B, C)
	$C = \cap A$	D = IN (1, A, C)
3) SYMMETRIC DIFFERENCE	$C = A \Delta B$	D = SD (A, B, C)
	$C = \Delta A$	D = SD (1, A, C)
4) RELATIVE COMPLEMENT	$C = A \sim B$	D = RL (A, B, C)
5) EXACTLY N ELEMENTS OF A	$C = E_n A$	D = EX (N, A, C)
6) DOMAIN of A	$C = \mathfrak{D}(A)$	D = DM (A, C)
7) RANGE of A	$C = \mathfrak{R}(A)$	D = RG (A, C)
8) IMAGE of B under A	C = A[B]	D = IM (A, B, C)
9) CONVERSE IMAGE under A	C = [B]A	D = CM (A, B, C)
10) CONVERSE of A	$C = \overline{A}$	D = CV (A, C)
11) RESTRICTION of A to B	C = A|B	D = RS (A, B, C)
12) RELATIVE PRODUCT of A and B	C = A/B	D = RP (A, B, C)
13) CARTESIAN PRODUCT of A and B	$C = A \times B$	D = XP (A, B, C)
14) DOMAIN CONCURRENCE of A to B	$C = \mathfrak{D}$ (A:B)	D = DC (A, B, C)
15) RANGE CONCURRENCE of A to B	$C = \mathfrak{C}$ (A:B)	D = RC (A, B, C)
16) SET CONCURRENCE of A to B	$C = \mathfrak{S}$ (A:B)	D = SC (A, B, C)
17) CARDINALITY of A N = $\aleph$ A, (N is an integer)		N = C (A)
BOOLEAN OPERATIONS	I = 1 if the statement is true. I = 0 if the statement is false.	
18) A is a subset of B		I = SBS (A, B)
19) A is equal to B		I = EQL (A, B)
20) A and B are disjoint		I = DSJ (A, B)
21) A is equipollent to B		I = EQP (A, B)
22) A is an element of B		I = ELM (A, B)
SPECIAL CONTROL OPERATIONS		
23) SET CONSTRUCTION	C = {A, B, X, . . .}	D = S(C, A, B, X, . . .)
24) MODE of A (see text) N is an integer		N = M(A)
25) ACCESS DATA in A by format N	(each format is written in the parent language and given an integer name)	D = ACC (N, A, C)

TABLE 1—Some set operations expressed as subroutines

REFERENCES

1 D L CHILDS
Feasibility of a set-theoretic data structure—a general structure based on a reconstituted definition of relation
IFIP Congress 68

2 T E JOHNSON
A mass storage relational data structure for computer graphics and other arbitrary data stores
MIT Department of Architecture Report October 1967

3 T SKOLEM
Two remarks on set theory
MATH SCAND 5 43–46 1957

4 P SUPPES
Axiomatic set theory
Van Nostrand Princeton 1960