TECHNICAL REPORT TR-179
VGL 21-002-008 AND
N00014-67-A-0239-0021
(NR-044-431)

MARCH 1972

F G R A A L

FORTRAN EXTENDED GRAPH ALGORITHMIC LANGUAGE

BY

V. R. BASILI
C. K. MESZTENYI
W. C. RHEINBOLDT

UNIVERSITY OF MARYLAND
COMPUTER SCIENCE CENTER
COLLEGE PARK, MARYLAND

TECHNICAL REPORT TR-179 MARCH 1972
NGL 21-002-008 AND
N00014-67-A-0239-0021
(NR-044-431)

F G R A A L

FORTRAN EXTENDED GRAPH ALGORITHMIC LANGUAGE

BY

V. R. BASILI
C. K. MESZTENYI
W. C. RHEINBOLDT

THIS RESEARCH WAS SUPPORTED IN PART BY THE OFFICE OF NAVAL RESEARCH AND THE NATIONAL AERONAUTICS AND SPACE ADMINISTRATION UNDER GRANT N00014-67-A-0239-0021(NR-044-431) AND GRANT NGL 21-002-008, RESPECTIVELY.

ABSTRACT

THIS REPORT DESCRIBES THE FORTRAN VERSION FGRAAL OF THE GRAPH ALGORITHMIC LANGUAGE GRAAL (TECHN.REPORT TR-158) AS IT HAS BEEN IMPLEMENTED FOR THE UNIVAC 1108. FGRAAL IS AN EXTENSION OF FORTRAN V. AND IS INTENDED FOR DESCRIBING AND IMPLEMENTING GRAPH ALGORITHMS OF THE TYPE PRIMARILY ARISING IN APPLICATIONS. THE FORMAL DESCRIPTION CONTAINED IN THIS REPORT REPRESENTS A SUPPLEMENT TO THE FORTRAN V. MANUAL FOR THE UNIVAC 1108 (UP-4060), THAT IS, ONLY THE NEW FEATURES OF THE LANGUAGE ARE DESCRIBED. SEVERAL TYPICAL GRAPH ALGORITHMS, WRITTEN IN FGRAAL, ARE INCLUDED TO ILLUSTRATE VARIOUS FEATURES OF THE LANGUAGE AND TO SHOW ITS APPLICABILITY.

TABLE OF CONTENTS

1. INTRODUCTION

A GRAPH ALGORITHMIC LANGUAGE, FGRAAL, IS DESCRIBED AS IT IS IMPLEMENTED FOR THE UNIVAC 1108. THE LANGUAGE IS AN EXTENSION OF FORTRAN V., AND IT IS INTENDED FOR IMPLEMENTING GRAPH ALGORITHMS OF THE TYPE PRIMARILY ARISING IN APPLICATIONS. THIS REPORT CONTAINS THE DESCRIPTION OF THE LANGUAGE EXTENSION FOR A POTENTIAL USER, AND IT SHOULD BE USED TOGETHER WITH THE FORTRAN V. MANUAL FOR THE UNIVAC 1108 (UP-4060). FGRAAL HAS BEEN IMPLEMENTED BY MODIFYING THE RALPH COMPILER. IT CAN BE CALLED BY

@FGRAAL,<OPTIONS> <SPECIFICATIONS>

WHERE <OPTIONS> AND <SPECIFICATIONS> ARE AS DEFINED FOR FORTRAN PROGRAMS.

CHAPTER 2 GIVES A SUMMARY OF THE SET THEORETIC FOUNDATION OF GRAAL. CHAPTERS 3-14 CONTAIN THE DESCRIPTION OF THE SPECIAL FEATURES OF THE LANGUAGE. CHAPTER 15 SUMMARIZES THE SPECIAL OPERATIONS AND STATEMENTS IN A TABLE. CHAPTER 16 CONTAINS SEVERAL EXAMPLES PROGRAMMED IN FGRAAL.

2. SET THEORETIC FOUNDATIONS OF GRAAL

IN THIS CHAPTER, CAPITAL LETTERS X,S,T, ETC. STAND FOR FINITE SETS, AND THE BASIC SET OPERATIONS ARE INDICATED BY THE USUAL SYMBOLS: $\cup$ (UNION), $\cap$ (INTERSECTION), $\sim$ (DIFFERENCE), AND Δ (SYMMETRIC SUM). FOR ANY SET X, THE CARDINALITY IS DENOTED BY $|X|$, P(X) IS THE POWER SET, AND WE DEFINE

$$P_k(X) = \{ S \in P(X) \mid |S| = k \}, \quad k = 0,1,\ldots,|X| \tag{2.1}$$

IN FGRAAL, THE SET OPERATIONS ARE DENOTED BY .UN., .IT., .DF. AND .SM., RESPECTIVELY. THE CARDINALITY OF A SET X IS GIVEN BY THE BUILT-IN FUNCTION SIZE(X).

IT IS WELL-KNOWN THAT UNDER UNION, INTERSECTION, AND COMPLEMENTATION (IN X), P(X) IS A BOOLEAN ALGEBRA WITH THE MEMBERS OF $P_1(X)$ AS GENERATORS. TO OBTAIN ANOTHER ALGEBRAIC STRUCTURE, LET GF(2) BE THE BINARY GALOIS FIELD WITH THE INTEGERS 0,1 AS ELEMENTS. THEN P(X) BECOMES A VECTOR SPACE OVER GF(2) IF THE SYMMETRIC SUM IS USED AS ADDITION AND THE SCALAR PRODUCT IS DEFINED BY

$$\lambda S = \emptyset \text{ FOR } \lambda = 0 \qquad \text{AND} \qquad \lambda S = S \text{ FOR } \lambda = 1$$

THE ELEMENTS OF $P_1(X)$ NOW FORM A BASIS.

FOR ANY SETS X,Y WE DENOTE BY B(X,Y) THE CLASS OF ALL MORPHISMS

$$\psi : P(X) \longrightarrow P(Y)$$

BETWEEN THE BOOLEAN ALGEBRAS P(X), P(Y). ANY $\psi \in B(X,Y)$ IS UNIQUELY CHARACTERIZED BY THE IMAGE SETS $\psi\{x\} \in P(Y)$ OF THE GENERATORS $\{x\} \in P_1(X)$ AND

$$\psi S = \bigcup_{x \in S} \psi\{x\}, \quad \forall S \in P(X) \tag{2.2}$$

CORRESPONDINGLY, WE DEFINE L(X,Y) AS THE CLASS OF ALL LINEAR MAPPINGS

$$\varphi : P(X) \longrightarrow P(Y)$$

BETWEEN THE VECTOR SPACES P(X), P(Y). THEN, INSTEAD OF (2.2), WE HAVE FOR ANY $\varphi \in L(X,Y)$ THE REPRESENTATION

$$\varphi S = \bigtriangleup_{x \in S} \varphi\{x\}, \quad \forall S \in P(X) \tag{2.3}$$

LET G BE A GRAPH WITH NODE SET V AND ARC SET A. THE ELEMENTS OF P(V) AND P(A) CONSTITUTE THE BASIC DATA OBJECTS FOR ALL OPERATIONS ON G UNDER GRAAL AND THE STRUCTURE OF THE GRAPH IS DEFINED BY CERTAIN BOOLEAN OR LINEAR MAPPINGS BETWEEN THE TWO POWER SETS. IN THE REMAINDER OF THIS CHAPTER WE DEFINE THE BASIC GRAPH OPERATORS PRESENTLY INCLUDED IN GRAAL.

IN FGRAAL, THE NODE SET V AND THE ARC SET A OF A GRAPH G ARE REFERED BY THE BUILT-IN FUNCTIONS NODES(G) AND ARCS(G), RESPECTIVELY.

AN UNDIRECTED PSEUDOGRAPH IS A TRIPLE $G = (V,A,\phi)$ CONSISTING OF A NODE SET V, AN ARC SET A, AND AN INCIDENCE OPERATOR

(2.4) (i) $\quad \phi \in B(A,V)$

(ii) $\quad \phi\{a\} \in P_1(V) \cup P_2(V), \qquad \forall_{a \in A}$

THUS FOR ANY ARC a, $\phi\{a\}$ IS EITHER THE TWO-ELEMENT SUBSET OF V CONSISTING OF THE TWO DISTINCT ENDPOINTS OF a, OR AN ATOMIC SUBSET OF V, IN WHICH CASE a IS A SELF LOOP. WE SPEEK OF A MULTIGRAPH IF IN (2.4) THE CONDITION (ii) IS REPLACED BY

(ii') $\quad \phi\{a\} \in P_2(V), \qquad \forall_{a \in A}$

THE UNQUALIFIED TERM GRAPH IS USED IF, IN ADDITION TO (ii'), THE RESTRICTED MAPPING $\phi : P_1(A) \rightarrow P_2(V)$ IS ONE-TO-ONE.

IN FGRAAL, THE INCIDENCE OPERATOR IS IMPLEMENTED BY A BUILT-IN FUNCTION, INC(G,S), WITH TWO ARGUMENTS, THE FIRST BEING THE GRAPH IDENTIFIER, AND THE SECOND A SET OF ARCS IN G. THE VALUE OF THE FUNCTION IS A SET (OF NODES OF G) AS DEFINED BY (2.2) AND (2.4).

FOR ANY UNDIRECTED PSEUDOGRAPH $G = (V,A,\phi)$ THE STAR OPERATOR IS THE BOOLEAN MAPPING

(2.5) $\quad \sigma \in B(V,A)$

$$\sigma\{v\} = \{a \in A \mid v \in \phi\{a\}\}, \quad \forall_{v \in V}$$

WHILE THE STANDARD BOUNDARY OPERATOR ∂ AND COBOUNDARY OPERATOR δ ARE DEFINED AS THE LINEAR MAPPINGS

(2.6) $\quad \partial \in L(A,V)$

$$\partial\{a\} = (|\phi\{a\}| - 1)\,\phi\{a\}, \quad \forall_{a \in A}$$

AND

(2.7) $\quad \delta \in L(V,A)$

$$\delta\{v\} = \{a \in \sigma\{v\} \mid |\phi\{a\}| = 2\}, \quad \forall_{v \in V}$$

HENCE, FOR ANY NODE v, $\sigma\{v\}$ IS THE SET OF ALL ARCS OF G WHICH ARE INCIDENT WITH v. THE BOUNDARY OPERATOR ∂ MAPS EACH ARC INTO THE SET OF ITS TWO ENDPOINTS, PROVIDED THEY ARE DISTINCT, AND OTHERWISE INTO THE EMPTY SET. FINALLY, $\delta\{v\}$ CONSISTS OF ALL ARCS INCIDENT WITH v EXCLUDING ANY SELF LOOPS.

IN FGRAAL, THE ABOVE OPERATORS ARE IMPLEMENTED AS BUILT-IN FUNCTIONS, STAR(G,T), BD(G,S) AND COB(G,T), EACH WITH TWO ARGUMENTS. THE FIRST IS THE GRAPH IDENTIFIER, AND THE SECOND IS A SET, T OR S, WHERE T IS A SUBSET OF NODES OF G, AND S A SUBSET OF ARCS OF G. THE VALUES OF THE FUNCTIONS ARE SETS OF NODES OR ARCS OF G ACCORDING TO THE DEFINITIONS (2.2) AND (2.5) FOR STAR, (2.3) AND (2.6) FOR BD, (2.3) AND (2.7) FOR COB.

IN A GRAPH G EACH ARC IS UNIQUELY DETERMINED BY THE TWO ELEMENT SET OF ITS ENDPOINTS, AND HENCE THE ARCS ARE LOSING SOME OF THEIR OWN IDENTITY. ACCORDINGLY, IT IS OFTEN EXPEDIENT TO WORK EXCLUSIVELY WITH THE NODES. FOR THIS WE INTRODUCE FOR A GRAPH $G = (V,A,\phi)$ THE <u>ADJACENCY OPERATOR</u>

(2.8) $$\alpha \in B(V,V)$$

$$\alpha\{v\} = \{u \in V \mid \exists\, a \in \sigma\{v\},\ \phi\{a\} = \{u,v\}\},\quad \forall_{v \in V}$$

THUS, α PRODUCES FOR EACH NODE v THE SET OF ALL NODES u WHICH FORM WITH v THE (DISTINCT) ENDPOINTS OF SOME ARC OF G.

IN FGRAAL, THE ADJACENCY OPERATOR IS IMPLEMENTED AS A BUILT-IN FUNCTION, ADJ(G,S), WITH THE GRAPH IDENTIFIER G AGAIN AS FIRST ARGUMENT AND THE SET OF NODES OF G AS THE SECOND. THE VALUE OF THE FUNCTION IS A SET OF NODES AS DEFINED BY (2.2) AND (2.8).

OTHER OPERATORS ARE POSSIBLE AND MAY BE INCLUDED LATER. EACH ONE OF THESE OPERATORS CAN BE USED IN PLACE OF ϕ TO CHARACTERIZE GRAPHS OF SPECIFIC TYPE. FOR EXAMPLE, NOTE THAT α HAS THE PROPERTIES

(2.9) (i) $\alpha \in B(V,V)$

(ii) $v \notin \alpha\{v\},\quad \forall_{v \in V}$

(iii) $u \in \alpha\{v\}$ IF AND ONLY IF $v \in \alpha\{u\}, \forall_{v \in V}$

IF V IS ANY SET AND THE MAPPING α SATISFIES (2.9) THEN $G = (V,A,\phi)$ WITH

$$A = \{\{u,v\} \in P_2(V) \mid u \in \alpha\{v\}\}$$

$$\phi \in B(A,V),\ \phi\{a\} = \{u,v\} \text{ IF } a = \{u,v\}, \forall_{a \in A}$$

IS A WELL-DEFINED GRAPH WITH α AS ITS ADJACENCY OPERATOR. WE CALL (V,α) THE NODE FORM REPRESENTATION OF G.

THE DEFINITION OF THE VARIOUS OPERATORS ARE EASILY CARRIED OVER TO DIRECTED GRAPHS. A DIRECTED PSEUDOGRAPH SHALL BE A QUADRUPLE $G = (V,A,\phi_+,\phi_-)$ CONSISTING OF A NODE SET V, AN ARC SET A, AS WELL AS A POSITIVE AND A NEGATIVE INCIDENCE OPERATOR

(2.10)
$$\phi_+, \phi_- \in B(A,V)$$
$$\phi_+\{a\}, \phi_-\{a\} \in P_1(V), \qquad \forall a \in A$$

IN OTHER WORDS, $\phi_+\{a\}$ AND $\phi_-\{a\}$ ARE ATOMIC SUBSETS OF P(V) CONSISTING OF THE INITIAL AND TERMINAL NODES OF a, RESPECTIVELY. IN MANY CASES, IT IS CONVENIENT TO USE THE COMBINED INCIDENCE OPERATOR

(2.11)
$$\phi \in B(A,V)$$
$$\phi\{a\} = \phi_+\{a\} \cup \phi_-\{a\}, \qquad \forall a \in A$$

AS IN THE UNDIRECTED CASE WE SPEAK OF A DIRECTED MULTIGRAPH IF $\phi\{a\} \in P_2(V)$, FOR ALL $a \in A$, AND OF A DIRECTED GRAPH (DIGRAPH) IF, IN ADDITION, $\phi : P_1(A) \longrightarrow P_2(V)$ IS ONE-TO-ONE.

THE POSITIVE AND NEGATIVE STAR OPERATORS OF THE DIRECTED PSEUDOGRAPH G ARE DEFINED BY

(2.12)
$$\sigma_+ \in B(V,A)$$
$$\sigma_+\{v\} = \{a \in A \mid v = \phi_+\{a\}\}, \qquad \forall v \in V$$
$$\sigma_- \in B(V,A)$$
$$\sigma_-\{v\} = \{a \in A \mid v = \phi_-\{a\}\}, \qquad \forall v \in V$$

AND WE INTRODUCE ALSO THE COMBINED STAR OPERATOR

(2.13)
$$\sigma \in B(V,A)$$
$$\sigma\{v\} = \sigma_+\{v\} \cup \sigma_-\{v\}, \qquad \forall v \in V$$

THUS, $\sigma_+\{v\}$ CONSISTS OF ALL THE ARCS BEGINNING AT v AND $\sigma_-\{v\}$ OF THOSE TERMINATING AT THAT NODE.

THE POSITIVE AND NEGATIVE BOUNDARY AND COBOUNDARY OPERATORS OF A DIRECTED PSEUDOGRAPH ARE NOW THOSE LINEAR MAPPINGS WHICH COINCIDE WITH THE INCIDENCE AND STAR OPERATORS ON THE APPROPRIATE FAMILY OF ATOMIC SETS:

(2.14)
$$\partial_+, \partial_- \in L(A,V)$$
$$\partial_+\{a\} = \phi_+\{a\}, \quad \partial_-\{a\} = \phi_-\{a\}, \qquad \forall a \in A$$

$$\delta_+ , \delta_- \in L(V,A)$$

$$\delta_+\{v\} = \sigma_+\{v\}, \quad \delta_-\{v\} = \sigma_-\{v\}, \quad \forall_{v\in V}$$

IT IS NATURAL TO DEFINE ALSO THE COMBINED MAPPINGS

(2.15)
$$\partial \in L(A,V)$$

$$\partial\{a\} = \partial_+\{a\} \Delta \partial_-\{a\}, \quad \forall_{a\in A}$$

$$\delta \in L(V,A)$$

$$\delta\{v\} = \delta_+\{v\} \Delta \delta_-\{v\}, \quad \forall_{v\in V}$$

THUS $\partial\{a\}$ IS AGAIN THE SET OF THE ENDPOINTS OF a IF THESE ENDPOINTS ARE DISTINCT, AND THE EMPTY SET, IF THEY ARE NOT. SIMILARLY, $\delta\{v\}$ IS ONCE MORE THE SET OF ALL ARCS INCIDENT WITH v EXCLUDING ALL SELF LOOPS.

FINALLY, WE DEFINE FOR A DIGRAPH $G = (V,A,\phi_+,\phi_-)$ THE POSITIVE AND NEGATIVE ADJACENCY OPERATORS BY THE RELATIONS

(2.16)
$$\alpha_+ , \alpha_- \in B(V,V)$$

$$\alpha_+\{v\} = \{u\in V \mid \exists\, a \in \sigma_+\{v\},\ u = \phi_-\{a\}\}, \forall_{v\in V}$$

$$\alpha_-\{v\} = \{u\in V \mid \exists\, a \in \sigma_-\{v\},\ u = \phi_+\{a\}\}, \forall_{v\in V}$$

THEN THE COMBINED ADJACENCY OPERATOR

(2.17)
$$\alpha \in B(V,V)$$

$$\alpha\{v\} = \alpha_+\{v\} \cup \alpha_-\{v\}; \quad \forall_{v\in V}$$

HAS AGAIN EXACTLY THE SAME MEANING AS IN THE UNDIRECTED CASE. MOREOVER, α_+ AND α_- MAY BE USED TO CHARACTERIZE THE INCIDENCE STRUCTURE OF A DIGRAPH. HERE CONDITIONS (i) AND (ii) OF (2.9) HAVE TO HOLD FOR BOTH α_+ AND α_-, AND (iii) IS REPLACED BY

(iii') $u \in \alpha_+\{v\}$ IF AND ONLY IF $v \in \alpha_-\{u\}, \forall u,v \in V$

THIS DEFINES THE NODE REPRESENTATION (V,α_+,α_-) OF A DIGRAPH.

IN FGRAAL, THE POSITIVE (NEGATIVE) OPERATORS ARE IMPLEMENTED BY HAVING THE LETTER 'P' ('N') IN FRONT OF THE CORRESPONDING BUILT-IN FUNCTION NAME. FOR EXAMPLE, THE POSITIVE INCIDENCE OPERATOR CORRESPONDS TO THE BUILT-IN FUNCTION PINC(G,S), THE NEGATIVE INCIDENCE OPERATOR TO NINC(G,S).

3. VARIABLES, DECLARATION STATEMENTS

THERE ARE FOUR NEW TYPES OF VARIABLES INTRODUCED IN FGRAAL; NAMELY SET-, LIST-, PROPERTY- AND GRAPH-VARIABLES. ALL OF THESE VARIABLES MUST APPEAR IN THE PROPER DECLARATION STATEMENT. SETS CONSTITUTE A NEW BASIC DATA TYPE. LISTS ARE DOUBLY-OPEN LINKED LIST STRUCTURE WHICH MAY BE USED AS STACKS OR QUEUES. PROPERTY VARIABLES RESEMBLE SUBSCRIPTED VARIABLES WITH SET VARIABLES CORRESPONDING TO SUBSCRIPTS. GRAPHS REPRESENT SPECIFIC DATA STRUCTURES. ALL OF THESE VARIABLES MAY APPEAR IN THE ARGUMENT LIST OF A SUBROUTINE CALL.

3.1. SETS

IN FGRAAL, SETS CONSTITUTE A NEW BASIC DATA TYPE. AN ATOMIC SET IS A SET CONSISTING OF ONE ITEM, AND ANY SET IS EITHER EMPTY OR A UNION OF ATOMIC SETS. THE ONLY CONSTANT SET IS THE EMPTY SET. AN ATOMIC SET CARRIES A SEQUENCE NUMBER WHICH IS ASSIGNED TO IT AT THE TIME OF ITS CREATION, AND THE MEMBERS OF A SET ARE ORDERED IN ASCENDING ORDER OF THEIR SEQUENCE NUMBER. ALL SEQUENCE NUMBERS ARE RETAINED IN AN ELEMENT SEQUENCE; IT IS AN ORDERED INTERNAL STRUCTURE SERVING THE DUAL PURPOSE OF CATALOGING THE ATOMIC SETS WHICH HAVE BEEN CREATED SO FAR AND OF PROVIDING THE LINKAGE BETWEEN AN ATOMIC SET AND THE PROPERTIES WHICH ARE ASSIGNED TO IT.

SET VARIABLES ARE DECLARED BY A SET DECLARATION, WHICH CONSISTS OF THE WORD SET FOLLOWED BY THE LIST OF VARIABLES, E.G.

```
SET  S,T
```

SETS CAN ALSO BE DECLARED IMPLICITLY, AS ANY OTHER VARIABLES IN FORTRAN. THE STATEMENT,

```
IMPLICIT SET (S-W)
```

CAUSES ALL VARIABLES WHOSE LEADING CHARATER IS IN THE RANGE FROM S TO W TO BE CONSIDERED AS SET VARIABLES.

THE CREATION OF ATOMIC SETS IS ACCOMPLISHED BY THE CREATE FUNCTION WHICH IS DESCRIBED IN THE SECTION 4.1.

THE EMPTY SET IS DENOTED BY THE CHARACTER '&' OR BY THE KEYWORD .EMPTY. ENCLOSED IN PERIODS. A DECLARED SET CAN BE MADE EMPTY BY THE FOLLOWING SET ASSIGNMENT STATEMENT:

```
S = &      OR     S = .EMPTY.
```

AND CAN BE TESTED IF IT IS EMPTY OR NOT BY THE LOGICAL EXPRESSION

S .EQ. & OR S .NE. &

SET VARIABLES MAY BE DIMENSIONED AS OTHER TYPE OF VARIABLES (REAL, INTEGER, ETC.), EITHER IN A SEPARATE DIMENSION STATEMENT OR IN THE DECLARATION STATEMENT ITSELF, E.G.

SET R(10)

DEFINES AN ARRAY CONSISTING OF 10 SETS, R(1),...,R(10).

3.2. LISTS

A LIST IS A DOUBLY-OPEN, LINKED LIST STRUCTURE WHICH MAY BE USED AS A STACK OR A QUEUE. IT OFFERS A DYNAMIC ALTERNATIVE TO THE ARRAY FOR THE STORING VALUES OF VARIABLES IN THE ORDER DEFINED BY THE USER. EACH LIST MUST BE DECLARED BY ITS TYPE (INTEGER, REAL, BOOLEAN OR SET) REFERING TO THE TYPE OF VALUES THE LIST MAY CONTAIN. THE LIST DECLARATION STATEMENT CONSISTS OF THE TYPE, FOLLOWED BY THE WORD STAQUE, FOLLOWED BY A LIST OF NAMES, E.G.

INTEGER STAQUE I,J

OPTIONALLY, THE TYPE MAY BE DELETED, IN WHICH CASE, THE TYPE OF THE LIST IS DEFINED BY THE 'NAME RULE'.

3.3. PROPERTIES

A PROPERTY MAY BE ASSOCIATED WITH ANY ATOMIC SET. THE PROPERTY DECLARATION ESTABLISHES THE TYPE ASSOCIATED WITH EACH PROPERTY NAME. THE DECLARATION CONSISTS OF THE TYPE, FOLLOWED BY THE WORD PROPERTY, FOLLOWED BY THE LIST OF PROPERTY NAMES, E.G.

```
SET PROPERTY  S,TREE
REAL PROPERTY  CAP,F1,F2
```

WHENEVER THE TYPE IS NOT EXPLICITLY GIVEN IN THE DECLARATION, THE TYPES OF THE PROPERTIES IN THE LIST ARE DEFINED BY THE 'NAME RULE'.

ASSIGNING OR REASSIGNING OF A PROPERTY VALUE TO AN ATOMIC SET , AND RETRIEVING IT, IS DESCRIBED IN CHAPTER 7.

3.4. GRAPHS

GRAPHS REPRESENT SPECIFIC DATA STRUCTURES TOGETHER WITH CERTAIN OPERATIONS FOR MANIPULATING THEM. THE GRAPH DECLARATION IDENTIFIES THE DATA STRUCTURES USED AND THE FAMILY OF OPERATORS AVAILABLE WITH IT. THE LANGUAGE IS MODULAR IN THE SENSE THAT, IN GENERAL, ONLY SOME OF THE POSSIBLE GRAPH OPERATIONS ARE USABLE WITH ANY SPECIFIC GRAPH. FOUR MODULES ARE PRESENTLY DEFINED IN THE LANGUAGE :

0	DIRECTED PSEUDOGRAPH
1	UNDIRECTED PSEUDOGRAPH
2	DIRECTED NODE GRAPH
3	UNDIRECTED NODE GRAPH

THE PSEUDOGRAPH IS THE MOST GENERAL MODULE AND ALLOWS SELF-LOOPS (ARC WITH THE SAME NODE AS END-POINTS) AND MULTIPLE ARCS (MORE THAN ONE ARE CONNECTING TWO DIFFERENT NODES). THE NODE GRAPH DOES NOT ALLOW SELF-LOOPS OR MULTIPLE ARCS, FURTHERMORE ARCS DO NOT HAVE AN IDENTITY.

THE IDENTITY OF A NODE OR ARC IS ESTABLISHED BY ASSIGNING ATOMIC SETS TO THE GRAPH. THESE OPERATIONS ARE DESCRIBED IN CHAPTER 9.

GRAPHS WITH DIFFERENT MODULES MUST BE DECLARED IN SEPARATE STATEMENTS. THE GRAPH DECLARATION STATEMENT CONSISTS OF THE WORD GRAPH, FOLLOWED BY THE GRAPH MODULE NUMBER IN PARENTHESES, AND BY A LIST OF GRAPH VARIABLES, E.G.

```
GRAPH(0) G,H
GRAPH(3) T
```

STATEMENTS DECLARE G AND H AS DIRECTED PSEUDOGRAPHS, T AS AN UNDIRECTED NODE GRAPH.

3.5. VARIABLES AS FUNCTION OR SUBROUTINE ARGUMENTS

ANY OF THE ABOVE DESCRIBED VARIABLES MY BE SUBMITTED TO FUNCTIONS OR SUBROUTINES IN THE ARGUMENT LIST, E.G.

```
SET  S, T(10)
GRAPH(1) G
REAL PROPERTY  LENGTH
     .........
CALL SUB(S,T,G,LENGTH)
     .........
```

WHEN A SUBROUTINE OR FUNCTION HAS THE ABOVE VARIABLES IN ITS ARGUMENT LIST, IT MUST DECLARE THEM, E.G.

```
SUBROUTINE SUB(S,T,G,LENGTH)
SET S, T(10)
GRAPH  G
REAL PROPERTY LENGTH
   .........
```

THE MODULE NUMBER OF THE GRAPH DECLARATION MAY BE OMITTED IN THE SUBROUTINE.

3.6. EXTERNAL SET FUNCTIONS

EXTERNAL FUNCTION SUBPROGRAMS OF SET TYPE CAN BE USED THE SAME WAY AS OTHER TYPE OF EXTERNAL FUNCTION SUBPROGRAMS, I.E. THE TYPE OF THE FUNCTION MUST BE DECLARED UNLESS ITS NAME IN CONTAINED IN AN IMPLICIT SET DECLARATION, E.G.

```
C      CALLING PROGRAM
          ......
       SET  SFCT
          ......
       X = ... SFCT(Z,R) ...
          ......
       END
-------------------------
C      FUNCTION PROGRAM
       SET FUNCTION SFCT(S,T)
          ......
       SFCT = ...
       RETURN
       END
```

4. SET EXPRESSION

A SET EXPRESSION IS A RULE FOR CREATING, REFERENCING, AND MANIPULATING SETS. EACH ATOMIC SET CARRIES A SEQUENCE NUMBER WHICH IS ASSIGNED TO IT AT THE TIME OF ITS CREATION. A SET IS A UNION OF ATOMIC SETS ORDERED IN ASCENDING ORDER BY THEIR SEQUENCE NUMBER. ALL SEQUENCE NUMBERS ASSIGNED TO ATOMIC SETS ARE RETAINED IN AN ELEMENT SEQUENCE. THIS IS AN ORDERED INTERNAL STRUCTURE SERVING THE DUAL PURPOSE OF CATALOGING THE ATOMIC SETS WHICH HAVE BEEN CREATED THUS FAR AND PROVIDING THE LINKAGE BETWEEN AN ATOMIC SET AND THE PROPERTIES WHICH ARE ASSIGNED TO IT.

4.1. CREATION AND REMOVAL OF ATOMIC SETS

THE SPECIAL FUNCTION,

CREATE (0)

WITH ARGUMENT ZERO CREATES A NEW ELEMENT WITH THE NEXT SEQUENCE NUMBER IN THE ELEMENT SEQUENCE AND RETURNS THE ATOMIC SET CONTAINING THAT ELEMENT. THE SAME FUNCTION MAY BE USED WITH A LIST OF ARGUMENT PAIRS:

CREATE (P1,V1,P2,V2,...,PN,VN) (N>0)

WHERE THE PI'S ARE DECLARED PROPERTY NAMES, AND THE VI'S ARE VARIABLES OR CONSTANTS OF THE SAME TYPES AS THE CORRESPONDING PROPERTIES. THIS VERSION OF THE FUNCTION CAUSES A SEARCH OF THE ELEMENT SEQUENCE FOR AN ATOMIC SET FOR WHICH ALL THE NAMED PROPERTIES EXIST AND ARE PRESENTLY ASSIGNED THE SPECIFIED VALUES. IF A (COMPLETE) MATCH IS FOUND THE FUNCTION RETURNS THE CORRESPONDING ATOMIC SET. IF NO MATCH (OR ONLY A PARTIAL MATCH) OCCURS, A NEW ELEMENT WITH THE NEXT SEQUENCE NUMBER IS ADDED TO THE ELEMENT SEQUENCE WITH THE GIVEN PROPERTIES AND CORRESPONDING VALUES. THE ATOMIC SET CONTAINING THIS NEW ELEMENT IS RETURNED.

THE REMOVE STATEMENT REMOVES ALL ATOMIC SETS WHICH ARE ELEMENTS OF THE ARGUMENT SET FROM THE ELEMENT SEQUENCE, TOGETHER WITH THEIR ASSOCIATED PROPERTIES, E.G.

REMOVE S

REMOVES THE ATOMIC SETS WHICH ARE ELEMENTS OF THE SET S. IF A REMOVED ATOMIC SET IS REFERENCED AN ERROR CONDITION OCCURS.

THE SPECIAL FUNCTION,

ATOM (I)

WHERE I IS AN INTEGER, CAN BE USED TO OBTAIN THE ATOMIC SET WHOSE ELEMENT HAS THE GIVEN SEQUENCE NUMBER I. IF THERE IS NO SUCH AN ELEMENT, IT RETURNS AN EMPTY SET.

4.2. SET OPERATORS

THE FOLLOWING FOUR SET OPERATORS ARE DEFINED IN FGRAAL :

.DF.	DIFFERENCE ($\sim$)
.UN.	UNION ($\cup$)
.IT.	INTERSECTION ($\cap$)
.SM.	SYMMETRIC SUM

THEY ARE LISTED IN INCREASING PRECEDENCE. S.DF.T GIVES THE SET CONSISTING OF THE ELEMENTS OF S WHICH ARE NOT IN T. S.UN.T GIVES THE SET CONSISTING OF ALL THE ELEMENTS OF S AND T, S.IT.T GIVES THE SET CONSISTING OF ALL ELEMENTS WHICH ARE ELEMENTS OF BOTH S AND T, S.SM.T GIVES THE SET OF ALL ELEMENTS WHICH ARE ELEMENTS OF S OR T BUT NOT BOTH.

4.3. SET FUNCTIONS

SET FUNCTIONS ARE BUILT-IN FUNCTIONS WHICH PRODUCE SETS. TWO OF THESE FUNCTIONS ARE THE CREATE AND ATOM FUNCTIONS DESCRIBED IN SECTION 4.1. OTHER SET PRODUCING FUNCTIONS ARE THE FUNCTIONS SUBSET AND EL1 (CHAPTER 8), THE GRAPH OPERATORS (CHAPTER 9), THE SET-PROPERTIES (CHAPTER 7) AND THE ELEMENTS OF A LIST OF TYPE SET (CHAPTER 6).

4.4. SET-EXPRESSIONS

SET EXPRESSIONS ARE FORMED BY THE PROPER SEQUENCE OF SETS, SET-OPERATORS, SET-FUNCTIONS AND PARENTHESES. THE PRECEDENCE OF THE SET OPERATORS AND FUNCTIONS ARE AS FOLLOWS (IN INCREASING ORDER) :

.DF.
.UN.
.IT.
.SM.
SET-FUNCTIONS

E.G. THE EXPRESSION

CREATE(0) .UN. S .SM. T .DF. R

WOULD BE EVALUATED AS THOUGH IT WERE PARENTHESIZED AS FOLLOWS

(CREATE(0) .UN. (S .SM. T)) .DF. R

WHEN TWO OPERATORS ARE OF THE SAME PRECEDENCE, THE ONE OF THE LEFT HAS PRECEDENCE OVER THE ONE ON THE RIGHT, E.G.

S .DF. T .DF. R

IS EVALUATED AS

(S .DF. T) .DF. R

PARENTHESES CAN BE USED TO ALTER THE PRECEDENCE RELATION.

IF A VARIABLE IS USED MORE THAN ONCE IN THE CONTEXT OF AN EXPRESSION OR ARGUMENT LIST, AND AT LEAST ONE SUCH A USE IS AS AN ARGUMENT OF A FUNCTION OR OPERATOR WHICH MODIFIES IT THEN CERTAIN PROBLEMS ARISE. SEE APPENDIX FOR MORE INFORMATION.

5. SET-RELATIONS, LOGICAL EXPRESSIONS

THE FOLLOWING 3 SET-RELATION OPERATORS MAY BE USED TO OBTAIN LOGICAL VALUE TRUE OR FALSE:

```
.EQ.
.NE.
.IN.
```

THE LOGICAL EXPRESSION

```
S.EQ.T
```

GIVES THE VALUE TRUE WHENEVER THE TWO SETS, S AND T, ARE EQUAL, AND THE VALUE FALSE OTHERWISE. THE LOGICAL EXPRESSION

```
S.NE.T
```

GIVES THE VALUE TRUE WHEN THE TWO SETS ARE NOT EQUAL, AND THE VALUE FALSE WHEN THEY ARE EQUAL. THE LOGICAL EXPRESSION

```
S.IN.T
```

GIVES THE VALUE TRUE WHEN THE SET S IS CONTAINED IN THE SET T (S C T), AND THE VALUE FALSE OTHERWISE.

THE ABOVE EXPRESSIONS MAY FORM A SUBEXPRESSION IN ANY FORTRAN TYPE OF LOGICAL EXPRESSION, E.G.

```
(I.LE.2).AND.(S.IN.T)
```

IF A VARIABLE IS USED MORE THAN ONCE IN THE CONTEXT OF AN EXPRESSION OR ARGUMENT LIST, AND AT LEAST ONE SUCH USE IS AS AN ARGUMENT OF A FUNCTION OR OPERATOR WHICH MODIFIES IT THEN CERTAIN PROBLEMS ARISE. SEE APPENDIX FOR MORE INFORMATION.

6. LIST OPERATIONS

AS STATED IN SECTION 3.2, A LIST IS BASICALLY A STACK OR A QUEUE. TO BUILD THE LIST, ITEMS ARE CONCATENATED TOGETHER. A LIST STATEMENT CONSISTS OF A LIST IDENTIFIER, THE EQUAL SIGN AND A LIST OF ITEMS SEPARATED BY COLONS. THE ITEMS MUST BE CONSTANTS, VARIABLES, LISTS BE USED FOR COLON. E.G.

```
REAL STAQUE  Q,R
Q = X : Q : R : 2.5
```

OR

```
Q = X .ET. Q .ET. R .ET. 2.5
```

LISTS APPEARING IN THE CONCATENATION ARE COPIED, E.G.

```
L = 1 : 2
LL = L : 3
```

IS EQUIVALENT TO

```
LL = 1 : 2 : 3
```

AN EMPTY LIST IS REFERENCED BY THE CHARACTER '#' OR BY .NIL. A LIST CAN BE EMPTIED BY THE ASSIGNMENT STATEMENT

```
L = #        OR        L = .NIL.
```

AND CAN BE TESTED BY

```
L .EQ. #     OR        L .EQ. .NIL.
```

TO OBTAIN AND POSSIBLY REMOVE AN ITEM FROM A LIST, THERE ARE FOUR OPERATORS :

```
FIRST (L)
```

RETURNS THE FIRST ITEM OF THE LIST L,

```
DFIRST (L)
```

RETURNS THE FIRST ITEM AND DELETES IT FROM THE LIST L,

```
LAST (L)
```

RETURNS THE LAST ITEM OF THE LIST L, AND

```
DLAST (L)
```

RETURNS THE LAST ITEM AND DELETES IT FROM THE LIST L.

THE ITEM RETURNED IS OF THE TYPE OF THE DECLARED LIST L, THUS IT MAY BE USED IN ANY PROPER CONTEXT (EXPRESSIONS).

IF A VARIABLE IS USED MORE THAN ONCE IN THE CONTEXT OF AN EXPRESSION OR ARGUMENT LIST, AND AT LEAST ONE SUCH USE IS AS AN ARGUMENT OF A FUNCTION OR OPERATOR WHICH MODIFIES IT THEN CERTAIN PROBLEMS ARISE. SEE APPENDIX FOR MORE INFORMATION.

7. PROPERTY ASSIGNMENT AND RETRIEVAL

AS STATED IN SECTION 3.3, PROPERTIES CAN BE ASSOCIATED WITH ATOMIC SETS. IN SECTION 4.1, PROPERTY ASSIGNMENT BY THE CREATE FUNCTION WAS ALREADY DESCRIBED.

A SIMPLE PROPERTY ASSIGNMENT IS DONE FOR THE ATOMIC SET X AS

```
P (X) = E
```

WHERE P IS A DECLARED PROPERTY AND E IS AN EXPRESSION OF THE SAME TYPE AS P (INTEGER, REAL, DOUBLE PRESICION, BOOLEAN OR SET). IF THE ATOMIC SET X HAD PROPERTY P ALREADY, THEN THE PREVIOUS VALUE IS RESET TO THE VALUE OF E. OTHERWISE, THE PROPERTY P IS ESTABLISHED WITH THE VALUE OF E.

WHEN P(X) IS REFERENCED, E.G. AS PART OF AN EXPRESSION, ITS CURRENT VALUE IS RETRIEVED. WHENEVER THE ARGUMENT SET X IS NOT AN ATOMIC SET, BUT A SET WITH MORE THAN ONE ELEMENT, THEN THE OPERATIONS ARE PERFORMED IN RESPECT TO THE FIRST ELEMENT OF THE SET.

THE SPECIAL BUILT-IN LOGICAL FUNCTION,

```
CHECK (P,X)
```

RETURNS THE VALUE TRUE WHEN THE PROPERTY P HAS BEEN DEFINED FOR THE ATOMIC SET X, AND RETURNS THE VALUE FALSE OTHERWISE.

THE REMOVE STATEMENT WITH ARGUMENT OF A PROPERTY NAME,

```
REMOVE P
```

REMOVES THE PROPERTY P FROM ALL ATOMIC SETS FOR WHICH IT HAD BEEN DEFINED, THE STATEMENT WITH ARGUMENT P FOLLOWED BY A SET IN PARENTHESES,

```
REMOVE P(X)
```

REMOVES THE PROPERTY P ONLY FROM THE ATOMIC SETS WHICH WERE ELEMENTS OF THE SET X.

8. BUILT-IN FUNCTIONS

8.1. ELT AND INDEX FUNCTIONS

THE FUNCTION,

ELT (I,S)

WITH INTEGER I AND SET S, RETURNS THE I-TH ELEMENT OF S AS AN ATOMIC SET. IF THERE IS NO I-TH ELEMENT OF S THEN IT RETURNS THE EMPTY SET.

THE FUNCTION,

INDEX (X,S)

WITH ATOMIC SET X AND SET S, RETURNS INTEGER I IF X IS THE I-TH ELEMENT OF S, RETURNS ZERO OTHERWISE.

8.2. SUBSET FUNCTION

THE FUNCTION,

SUBSET (X , B)

RETURNS A SET OF ELEMENTS WHICH SATISFY THE LOGICAL EXPRESSION B. X IS A DUMMY VARIABLE WHICH CAN BE USED IN THE LOGICAL EXPRESSION B.
E.G.

SUBSET(X , (X.IN.S).AND.(DIST(X).GT.0))

8.3. SIZE, PARITY, COUNT FUNCTIONS

THE FUNCTION,

SIZE (S)

RETURNS THE NUMBER OF ELEMENTS IN THE SET S.

THE FUNCTION,

PARITY (S)

RETURNS ZERO IF THE SIZE(S) OS EVEN, RETURNS ONE OTHERWISE.

THE FUNCTION,

COUNT (X)
OR COUNT (0)

RETURNS THE SEQUENCE NUMBER OF THE GIVEN ATOMIC SET X OR RETURNS THE SEQUENCE NUMBER OF THE LAST ATOMIC SET CREATED WHEN THE ARGUMENT IS ZERO.

8.4. DOMAIN FUNCTION

THE FUNCTION,

DOMAIN (P)

WHERE P IS A PROPERTY IDENTIFIER, GENERATES THE SET WHOSE ELEMENTS HAVE PROPERTY P DEFINED. THIS FUNCTION PROVIDES A FASTER OBJECT TIME EXECUTION THAN THE EQUIVALENT FUNCTIONAL EXPRESSION

SUBSET (X , CHECK(P,X))

9. GRAPH OPERATIONS

GRAPH OPERATIONS ALLOW FOR THE ASSIGNMENT OF ATOMIC SETS AS NODES OR ARCS TO A GRAPH, THE REMOVAL OF NODES AND ARCS FROM A GRAPH, AND THE OBTAINING OF SUBSETS OF NODES OR ARCS OF A GRAPH.

9.1. GRAPH ASSIGNMENT STATEMENTS

THE STATEMENT, ASSIGN, IS USED TO ASSIGN ATOMIC SETS AS NODES OR ARCS TO A GRAPH. THREE FORMS OF ASSIGNMENT IS POSSIBLE :

(I) ASSIGNMENT OF AN ISOLATED NODE :

```
ASSIGN G, X
```

THE ATOMIC SET X IS ASSIGNED AS A NODE TO THE GRAPH G.

(II) ASSIGNMENT OF AN ARC IN A NODE-GRAPH :

```
ASSIGN G, X-Y
```

THE ATOMIC SETS, X AND Y, ARE ASSIGNED AS NODES TO THE GRAPH G, AND THE CONNECTION OF THE TWO NODES IS KEPT INTERNALLY.

(III) ASSIGNMENT OF AN ARC IN A GRAPH :

```
ASSIGN G, X-Y, Z
```

THE ATOMIC SETS, X AND Y, ARE ASSIGNED AS NODES TO THE GRAPH G, THE ATOMIC SET, Z, IS ASSIGNED AS AN ARC TO G.

WHEN THE GRAPH HAD BEEN DECLARED AS A DIRECTED GRAPH, THE ARC ASSIGNMENT X-Y IS UNDERSTOOD AS FROM X TO Y.

IF EITHER OR BOTH OF THE NODES HAVE BEEN PREVIOUSLY ASSIGNED TO G, THEN JUST THE UNASSIGNED NODE (IF ANY) AND THE ARC WILL BE ASSIGNED TO G.

9.2. REMOVAL OF ARCS AND NODES

THE DETACH STATEMENT IS USED TO REMOVE ARCS AND NODES FROM A GRAPH. THE DETACH STATEMENT HAS 3 FORMS:

(I) EMPTYING A GRAPH

DETACH G

REMOVES ALL ELEMENTS, NODES AND ARCS, FROM GRAPH G.

(II) REMOVAL OF ELEMENTS OF A GRAPH

DETACH G, S

REMOVES THE ELEMENT OF SET S FROM THE GRAPH G IF THEY HAD BEEN ASSIGNED AS NODES OR ARCS TO G PREVIOUSLY. WHEN A NODE OF G IS REMOVED THEN ALL THE INCIDENT ARCS OF THE NODE ARE ALSO REMOVED.

(III) REMOVAL OF ARCS

DETACH G, S-T

REMOVES ALL ARCS FROM G WHICH CONNECT THE TWO SETS OF NODES, S AND T, IN THE GRAPH G. IF THE GRAPH IS DIRECTED, THIS WILL MEAN ARCS FROM NODES IN S TO NODES IN T.

9.3. NODES, AND ARCS OF A GRAPH

THE FUNCTIONS,

NODES (G)
ARCS (G)

RETURN SETS CONSISTING THE ELEMENTS WHICH WERE ASSIGNED TO G AS NODES AND ARCS, RESPECTIVELY.

9.4. INCIDENCE OPERATORS

THE INCIDENCE OPERATORS ARE BUILT-IN FUNCTIONS WITH ARGUMENTS CONSISTING OF A GRAPH AND A SET WHOSE ELEMENTS WERE ASSIGNED TO THE GRAPH AS NODES OR ARCS. THE VALUE OF THE FUNCTION IS A SET WHICH IS A SUBSET OF THE NODES OR ARCS OF THE GRAPH. THE FORM OF THE FUNCTIONS ARE AS FOLLOWS:

OP (G,S)

WHERE G IS THE GRAPH NAME, S IS A SET OF NODES OR ARCS OF G (S MAY BE A SET EXPRESSION). THE VARIOUS OPERATORS (OP) ARE AS FOLLOWS:

INC (G,A) — A MUST BE AN ARC-SET OF G. THE FUNCTION RETURNS A SET WHICH IS THE UNION OF ALL NODES WHICH ARE

BOUNDARY NODES OF THE ARCS IN A.

STAR (G,V) — V MUST BE A NODE-SET OF G. THE FUNCTION RETURNS A SET WHICH IS THE UNION OF ALL ARCS WHICH ARE INCIDENT TO THE NODES IN V.

BD (G,A) — A MUST BE AN ARC-SET OF G. THE FUNCTION RETURNS A SET WHICH IS THE SYMMETRIC SUM OF ALL NODES WHICH ARE BOUNDARY NODES OF THE ARCS IN A.

COB (G,B) — V MUST BE A NODE-SET OF G. THE FUNCTION RETURNS A SET WHICH IS THE SYMMETRIC SUM OF ALL ARCS WHICH ARE INCIDENT TO THE NODES IN V.

ADJ (G,V) — V MUST BE NODE-SET OF G. THE FUNCTION RETURNS A SET WHICH IS THE UNION OF ALL NODES WHICH ARE ADJACENT TO THE NODES IN V.

IN CASE OF A NODE GRAPH G, ONLY THE ADJ OPERATOR IS DEFINED.

IF G IS A DIRECTED GRAPH, ALL THE ABOVE OPERATORS HAVE EXTENDED OPERATORS BY PLACING P OR N IN FRONT OF THEIR NAMES :

INC	PINC	NINC
STAR	PSTAR	NSTAR
BD	PBD	NBD
COB	PCOB	NCOB
ADJ	PADJ	NADJ

THE P (POSITIVE) OPERATORS ARE DEFINED SUCH THAT FOR ARCS THEY GIVES THE STARTING NODES, FOR NODES THEY GIVE THE ARCS WHICH ARE DIRECTED OUTWARD FROM THE NODES.

THE N (NEGATIVE) OPERATORS ARE DEFINED SUCH THAT FOR ARCS THEY GIVE THE ENDING NODES, FOR NODES THEY GIVE THE INCOMING ARCS.

10. SET ASSIGNMENT STATEMENT

THE SET ASSIGNMENT STATEMENT IS SIMILAR TO THE FORTRAN ASSIGNMENT STATEMENT :

```
V = E
```

WHERE V MUST BE A SET VARIABLE, E MUST BE A SET EXPRESSION. SPECIAL CONVERSION TAKES PLACE IF V OR E IS INTEGER TYPE :

```
I = E
```

WHERE I IS AN INTEGER VARIABLE AND E IS A SET EXPRESSION, IS EQUIVALENT TO

```
I = INDEX (1,E)
```

THE ASSIGNMENT,

```
V = I
```

WHERE V IS A SET VARIABLE, IS EQUIVALENT TO

```
V = ATOM (I)
```

11. ITERATIVE STATEMENT

THERE ARE TWO NEW ITERATIVE STATEMENTS IN FGRAAL WHICH ARE SIMILAR TO THE DO STATEMENT IN FORTRAN.

11.1. WHILE STATEMENT

THE FORM OF THE WHILE STATEMENT IS AS FOLLOWS :

```
DO  ST.NO.  WHILE  LOG.EXPR.
```

THE STATEMENT CAUSES THE EXECUTION OF THE STATEMENT FOLLOWING THE WHILE STATEMENT INCLUDING THE STATEMENT WHICH HAS THE SPECIFIED STATEMENT NUMBER REPETITIVELY UNTIL THE LOGICAL EXPRESSION HAS THE VALUE FALSE. THE TEST IS DONE BEFORE THE EXECUTION OF THE CONSECUTIVE STATEMENTS.
E.G.

```
DO 10 WHILE S.NE.&
```

11.2. FORALL STATEMENT

THE FORM OF THE FORALL STATEMENT IS AS FOLLOWS :

```
DO  ST.NO. FORALL  V.IN.E
```

WHERE V MUST BE A SET VARIABLE, E IS A SET EXPRESSION. THE SET E SHOULD NOT BE ALTERED IN THE RANGE OF THE DO STATEMENT. THE STATEMENT IS EQUIVALENT TO

```
N = SIZE (E)
DO  ST.NO.  I = 1,N
V = ELT (I,E)
```

EXCEPT THAT THE LOOP MAY BE SKIPPED COMPLETELY WHEN THE SET E IS EMPTY.
E.G.

```
DO 15 FORALL  X.IN.T
```

12. REMOVE STATEMENT

THE REMOVE STATEMENT REMOVES ATOMIC SETS FROM THE UNIVERSAL SEQUENCE, OR REMOVES DEFINED PROPERTIES OF ATOMIC SETS. THE STATEMENT CONSISTS OF THE WORD 'REMOVE' FOLLOWED BY A LIST OF ARGUMENTS SEPARATED BY COMMAS:

```
REMOVE  S,T,...,P,Q,...,PP(X),PQ(Y),...
```

THE ARGUMENTS MAY BE:

(1) SETS - THEN ALL ATOMIC SETS WHICH ARE ELEMENTS OF THE SET ARE REMOVED FROM THE UNIVERSAL SEQUENCE;
(2) PROPERTY NAMES - THE PROPERTY IS REMOVED FROM ALL ATOMIC SETS WHERE IT HAD BEEN ASSIGNED;
(3) PROPERTY NAMES FOLLOWED BY SET IN PARENTHESES - THE PROPERTY IS REMOVED FROM THOSE ATOMIC SETS WHICH ARE ELEMENTS OF THE GIVEN SET.

13. SAVE, RESET STATEMENTS

THE SAVE STATEMENT IS USED TO TRANSFER INFORMATION FROM CORE TO AN AUXILIARY STORAGE FILE. RESET STATEMENT IS USED TO RETRIEVE INFORMATION FROM THE AUXILIARY STORAGE. THE INFORMATION CAN BE EITHER PROPERTY VALUES, OR A FULL GRAPH.

AT THE PRESENT, THESE STATEMENTS ARE NOT IMPLEMENTED.

14. SPECIAL LIBRARY SUBROUTINES

THE FOLLOWING THREE CLOSELY RELATED SUBROUTINES ARE AVALILABLE

```
CONTRT(G,S,X)
NODSET(G,X,S)
EXPAND(G,X)
```

WHICH CAN BE CALLED BY STANDARD FORTRAN CALL STATEMENTS.

SUBROUTINE CONTRT CONTRACTS A GIVEN SUBSET S OF NODES OF GRAPH G INTO A NEw NODE WHICH IS RETURNED IN THE ATOMIC SET X. ON SUBSEQUENT USAGE OF THE GRAPH, X IS TO BE CONSIDERED AS A NODE OF G, BUT THE ELEMENTS OF S WILL NOT BE NODES OF G. THE SUBROUTINE MAY BE CALLED SUBSEQUENTLY, I.E. ONE MAY FURTHER CONTRACT AN ALREADY CONTRACTED GRAPH. MODIFICATION OF A CONTRACTED GRAPH BY ASSIGN AND DETACH STATEMENTS SHOULD BE AVOIDED.

SUBROUTINE NODSET RETURNS A SET S CONSISTING OF THE PREVIOUS NODES OF G WHICH WERE CONTACTED TO THE NODE X.

SUBROUTINE EXPAND EXPANDS THE PREVIOUSLY CONTRACTED NODES OF GRAPH G CORRESPONDING TO THE NODE IN ATOMIC SET X. X WILL NOT BE A NODE OF GRAPH G UPON RETURN.

CARE SHOULD BE TAKEN BY THE PROPER USAGE OF THESE SUBROUTINES, I.E. EXPANSIONS OF A GRAPH SHOULD BE IN THE REVERSE ORDER OF ITS CONTRACTIONS.

15. SUMMARY OF OPERATORS AND STATEMENTS

CONSTANTS: & OR .EMPTY. = EMPTY SET
OR .NIL. = EMPTY LIST

SET OPERATORS.

NAME	MEANING OF R = S .NAME. T
.DF.	DIFFERENCE: R IS THE SET OF ALL ELEMENTS OF THE SET S WHICH ARE NOT ELEMENTS OF THE SET T.
.UN.	UNION: R IS THE SET OF ALL ELEMENTS OF SETS S AND T.
.IT.	INTERSECTION: R IS THE SET OF ALL ELEMENTS WHICH ARE ELEMENTS OF BOTH SETS, S AND T.
.SM.	SYMMETRIC SUM: R IS THE SET OF ALL ELEMENTS WHICH ARE ELEMENTS OF SETS S OR T, BUT NOT BOTH.
.EQ.	EQUAL RELATION: R IS TRUE IF SETS S AND T ARE EQUAL, FALSE OTHERWISE.
.NE.	NOT-EQUAL RELATION: R IS TRUE IF SETS S AND T ARE NOT EQUAL, FALSE OTHERWISE.
.IN.	CONTAINED RELATION: R IS TRUE IF SET S IS CONTAINED IN SET T, FALSE OTHERWISE.

FUNCTION TABLE.

NAME	NO. ARGS	ARGS.	MEANING	FCT TYPE
			--- LIST FUNCTIONS: ---	
FIRST DFIRST LAST DLAST	1	LIST	RETURNS THE FIRST OR LAST ELEMENT OF THE LIST, WITH 'D', IT ALSO DELETES THE ELEMENT FROM THE LIST.	TYPE OF LIST
			--- SET FUNCTIONS: ---	
CREATE	1	ZERO	CREATED ELEMENT AS AT.SET	SET
CREATE	2N	PROP.NAME AND VALUE IN PAIRS	GIVES AT.SET WITH MATCHING PROPERTIES, CREATES ONE IF NONEXISTENT.	SET
ATOM	1	INT.	AT.SET WITH THE GIVEN SEQ.NUMBER	SET
ELT	2	INT,SET	AT.SET IN SPEC. PLACE IN THE SET	SET
INDEX	2	AT.SET, SET	INDEX NO. OF AT.SET IN THE SET.	INT
SIZE	1	SET	NUMBER OF ELEMENTS IN THE SET	INT
PARITY	1	SET	TRUE FOR ODD, FALSE FOR EVEN NUMBER OF ELEMENTS.	LOG
COUNT	1	ZERO	MAXIMAL SEQUENCE NUMBER	INT
COUNT	1	AT.SET	SEQUENCE NUMBER OF THE AT.SET	INT
SUBSET	2	SET,LOG	ELEMENTS OF FIRST,DUMMY ARGUMENT WHICH SATISFY LOG.EXPR.	SET
CHECK	2	PROP.NAME AT.SET	TRUE IF PROPERTY IS DEFINED FOR AT.SET, OTHERWISE FALSE	LOG
			--- PROPERTY FUNCTIONS: ---	
'PROP NAME'	1	AT.SET	RETURNS THE ASSIGNED PROP.VALUE	TYPE OF PROP
			--- GRAPH FUNCTIONS: ---	
NODES	1	GRAPH	NODES OF THE GRAPH	SET
ARCS	1	GRAPH	ARCS OF THE GRAPH	SET
INC	2	GRAPH, SET	UNION OF ALL BOUNDARY NODES OF THE GIVEN SET OF ARCS.	SET
STAR	2	GRAPH, SET	UNION OF ALL ARCS INCIDENT TO THE GIVEN SET OF NODES.	SET
BD	2	GRAPH, SET	SYMMETRIC SUM OF ALL BOUNDARY NODES OF ARCS IN THE SET.	SET
COB	2	GRAPH, SET	SYMMETRIC SUM OF ALL ARCS INCIDENT TO THE NODES IN THE SET.	SET
ADJ	2	GRAPH, SET	UNION OF ALL NODES ADJACENT TO THE NODES IN THE SET.	SET

STATEMENTS

CATEG.	FORM	MEANING
DECLARATION		
	SET X,Y,...	X,Y,... ARE SET VARIABLES.
	'TYPE' STAQUE L,P,...	L,P,... ARE LISTS OF 'TYPE'= REAL, INTEGER, LOGICAL OR SET.
	'TYPE' PROPERTY A,B,...	A,B,... PROP.FUNCTIONS OF 'TYPE' REAL, INT., LOG. OR SET.
	GRAPH G('MOD'),T('MOD'),...	G,T,... ARE GRAPH OF 'MOD'= DIRECTED OR UNDIRECTED, AND PSEUDO-,MULTI- OR NODE-GRAPHS.
ASSIGNMENT		
	S='SET-EXPRESSION'	SET ASSIGNMENT
	L=...:X:L:Y:...	LIST ASG.: X,L,Y,... ARE CONST., VBLE. OR LIST OF SAME TYPE.
	P(X)='EXPRESSION'	PROPERTY 'P' ASSIGNED TO ATOMIC SET X WITH VALUE 'EXPRESSION'
GRAPH		
	ASSIGN G, X	AT.SET X ASSIGNED TO G AS NODE.
	ASSIGN G, X-Y	ATOMIC SETS X,Y ARE ASSIGNED AS ADJACENT NODES IN GRAPH G.
	ASSIGN G, X-Y, Z	AT.SETS X,Y,Z ASSIGNED TO GRAPH G AS ARC Z WITH END-NODES X,Y.
	DETACH G	GRAPH G MADE EMPTY
	DETACH G, S	ELEMENTS OF SET S ARE REMOVED FROM GRAPH G
	DETACH G, S-T	ARCS CONNECTING S AND T REMOVED
ITERATIVE		EXECUTES STATEMENTS THROUGH ONE LABELED WITH 'ST#'
	DO 'ST#' FORALL X,IN,S	FOR EACH ELEMENT X OF SET S
	DO 'ST#' WHILE LOG.EXPR.	WHILE LOG.EXPRESSION IS TRUE
REMOVE		
	REMOVE S,T,...,A,B,...	REMOVES SETS S,T FROM UNIVERSE, PROPERTIES A,B FROM AT.SETS WHERE DEFINED.
SAVE-RESET		GRAPHS G,..., PROPERTIES P,...
	SAVE (I) G,...,P,...	SAVED ON AUXILIARY STORAGE I
	RESET (I) G,...,P,...	RESET FROM AUXILIARY STORAGE.

16. EXAMPLES

IN THE FOLLOWING PAGES, SOME EXAMPLES ARE PRESENTED. ALL THE PROGRAMS ARE WRITTEN AS SUBROUTINES, THUS DECLARATIONS FOR THE VARIABLES APPEARING IN THE CALLING SEQUENCES ARE ASSUMED TO BE DONE IN A CALLING MAIN PROGRAM.
THE EXAMPLES ARE AS FOLLOWS:

1. READ I. READ DATA FOR GRAPH G
2. READ II. READ DATA FOR GRAPH G
4. SUBGRAPH. ESTABLISHING A GRAPH WHICH IS A SUBGRAPH OF G
5. LINE GRAPH. ESTABLISHING THE LINE GRAPH OF G
6. CONDENSE. ESTABLISHING A CONDENSE GRAPH OF G
7. COCYCLES. COCYCLES OF GRAPH G
8. SPANNING TREE. SPANNING TREE OF GRAPH G
9. CYCLES. CYCLES OF GRAPH G
10. CUTS. FUNDAMENTAL CUTS OF GRAPH G
11. SHORTEST PATH. 2-DIRECTIONAL SEARCH FOR SHORTES PATH IN GRAPH G

```
 *****
 TITLE
 *****
READ 1, READ DATA FOR GRAPH G
      :
 ************
 SPECIFICATION
 ************
THE SUBROUTINE ASSUMES THAT THE FIRST RECORD
FROM UNIT IU CONTAINS THE SIZES, N AND M, OF
THE NODE AND ARC SET, AND THAT THEN M RECORDS.
ARE SUPPLIED EACH CONTAINING THREE INTEGERS.
ANY SUCH TRIPLE (K,I,J) SATISFIES
                1 .LE. K .LE. M
                1 .LE. I .LE. N
                1 .LE. J .LE. N
AND SIGNIFIES THAT THE K'TH ARC HAS THE I'TH
NODE AS INITIAL, AND THE J'TH NODE AS TERMI-
NAL VERTEX.
      :
 **************
 SEQUENCE CHART
 **************
READ SIZES
      :
CREATE AN ELEMENT FOR EACH NODE AND ARC
      :
      :
LOOP TO READ TRIPLETS AND
*          TO ASSIGN THEM TO G
*          :
*          :
*          :
END OF LOOP AND
    PROGRAM
      :
      :
```

```
      SUBROUTINE RDONE(G,IU)


      GRAPH G




      SET X



      READ (IU) N,M

      DO 10 L=1,N+M
   10 X=CREATE(0)

      DO 20 L=1,M
      READ (IU) K,I,J
   20 ASSIGN G, ATOM(I)-ATOM(J), ATOM(K+N)

      RETURN
      END
```

```
 *****
 TITLE
 *****
READ II. READ DATA FOR GRAPH G.
     :
     :
 *************
 SPECIFICATION
 *************
THE SUBROUTINE READ(BUFFER) IS ASSUMED TO BE
AVAILABLE WHICH ALLOWS THE INPUT OF VARIABLE
LENGTH RECORD OF BCD WORDS INTO THE LIST BUFFER.
THE UNDIRECTED GRAPH IS REPRESENTED IN NODE FORM
AND IS READ-IN IN TERMS OF PATHS, THAT IS, AS
SEQUENCES OF NODES FORMING PATHS IN G. THE INPUT
IS TERMINATED WITH A RECORD CONTAINING THE SINGLE
WORD 'LAST'.
     :
     :
 **************
 SEQUENCE CHART
 **************
READ FIRST RECORD
     :
LOOP TO PROCESS RECORDS
*
*    GET ELEMENT WITH PROPER NAME
*         :
*** ILLEGAL COMMAND ON COL. 2 ***
*
*........ YES, ASSIGN ISOLATED NODE
*               :
*               :
*........ NO, PROCESS PATH
                :
                :
                :
                :
     READ NEXT PATH
          :
END OF LOOP AND PROGRAM
     :
     :
```

```
      SUBROUTINE RDTWO(G,NAME)


      GRAPH G
      REAL PROPERTY NAME


      STAQUE BUFFER
      SET X,Y


      CALL READ(BUFFER)

      DO 30 WHILE (FIRST(BUFFER) .NE. 'LAST')

      X = CREATE(NAME,DFIRST(BUFFER))

      IF (BUFFER .NE. #) GO TO 10

      ASSIGN G, X
      GO TO 30

   10 DO 20 WHILE (BUFFER .NE. #)
      Y = CREATE(NAME,DFIRST(BUFFER))
      ASSIGN G, X - Y
   20 X = Y

   30 CALL READ(BUFFER)

      RETURN
      END
```

```
*****
TITLE
*****
SUBGRAPH
     :
     :
*************
SPECIFICATION
*************
THE SUBROUTINE SETS UP THE SUBGRAPH OF G WHICH
HAS A GIVEN SET N OF NODES OF G AS A NODE SET.
     :
**************
SEQUENCE CHART
**************

LOOP TO PROCESS NODES X IN N
*
*
*    GET SET OF ARCS INCIDENT TO X WITH OTHER END NODE IN
*    N.
*         :
*         :
*    PROCESS S AS ARCS OF SUBG
*         IS X AN ISOLATED NODE
*         *
*         *... YES, ASSIGN IT
*         *         :
*         *         :
*         *... NO,
*              LOOP TO PROCESS THE ARCS IN S
*              *
*              *    GET OTHER END NODE, AND ASSIGN THEM
*              *         :
*              *         :
*              *         :
*              END OF LOOP FOR S
*                   :
END OF LOOP AND PROGRAM
     :
     :
```

```
      SUBROUTINE SBGRPH(G,N,SUBG)


      GRAPH G, SUBG
      SET N


      IMPLICIT SET(A-Z)


      DO 30 WHILE (N .NE. &)
      X = ELT(1,N)

      S = SUBSET(A,(A .IN. STAR(G,X)) .AND. (INC(G,A) .IN. N)))
      N = N.DF.X

      IF (S .NE. &) GO TO 10

      ASSIGN SUBG, X
      GO TO 30

10    DO 20 FORALL A.IN.S

      Y = INC(G,A).DF.X
      IF (Y.EQ.&) Y = X
20    ASSIGN SUBG, Y = Y, A

30    CONTINUE

      RETURN
      END
```

```
 *****
 TITLE
 *****
LINE GRAPH.
     :
 ************
 SPECIFICATION
 ************
THIS PROCEDURE SETS UP THE LINE GRAPH OF G, THAT IS,
THE GRAPH WHICH HAS THE ARCS OF G AS NODES AND IN WHICH
TWO NODES ARE ADJACENT WHENEVER THE CORRESPONDING ARCS ARE.
     :
 **************
 SEQUENCE CHART
 **************

LOOP FOR NODES OF G
*
*    GET INCIDENT ARCS OF NODE X
*         :
*         :
*    LOOP TO PROCESS THE ARCS
*    *
*    *    ASSIGN SELF-LOOP FOR SELF-LOOP
*    *         :
*    *         :
*    *    ASSIGN REST OF THE ARCS
*    *         :
*    *         :
END OF LOOPS
     :
     :
```

```
      SUBROUTINE LINEGR(G,LINEG)



      GRAPH G, LINEG



      IMPLICIT SET(A-Z)



      DO 10 FORALL X.IN.NODES(G)

      S = STAR(G,X)
      R = S

      DO 10 FORALL A.IN.S

      IF (X.IN.INC(G,A)) ASSIGN LINEG, A-A
      R = R .DF. A

      DO 10 FORALL B.IN.R
   10 ASSIGN LINEG, A-B

      RETURN
      END
```

```
 *****
 TITLE
 *****
CONDENSE GRAPH
     :
     :
     :
 **************
 SPECIFICATION
 **************
THE LIST L IS ASSUMED TO CONTAIN A FAMILY OF SETS
REPRESENTING A PARTITION OF THE NODE SET OF G. THE
PROCEDURE SETS UP A CONDENSED GRAPH WHICH HAS THE
MEMBERS OF L AS NODES AND IN WHICH TWO NODES ARE
ADJACENT IF THERE IS AT LEAST ONE ARC BETWEEN THE
CORRESPONDING SETS OF NODES IN G. THE PROPERTY REF
OF THE NODES OF CONG REMEMBERS THE SETS OF L.
     :
 ***************
 SEQUENCE CHART
 ***************

LOOP TO PROCESS THE SETS IN THE LIST
*
*
*
*
END OF LOOP
     :
LOOP TO GET ADJACENCY
*
*    FROM NODE X
*    LOOP TO OTHER NODES
*    *
*    *
*    *
*    END OF LOOP
*
END OF LOOPS
     :
     :
```

```
      SUBROUTINE CONDS(G,L,CONG,REF)



      GRAPH G, CONG
      SET STAQUE L
      SET PROPERTY REF









      IMPLICIT SET(A-Z)



      DO 10 WHILE (L.NE.#)
      X = CREATE(0)
      ASSIGN CONG, X
   10 REF(X) = DFIRST(L)

      R = &

      DO  30  FORALL  X .IN. NODES(CONG)
      T = ADJ(G,REF(X))

      DO  20  FORALL  Z .IN. R
      IF (T.IT.REF(Z) .NE. &) ASSIGN CONG, X - Z
   20 CONTINUE

   30 R = R .UN. X

      RETURN
      END
```

```
*****
TITLE
*****
COCYCLES.
     :
     :
************
SPECIFICATION
************
THIS SUBROUTINE DETERMINES A BASIS FOR THE COCYCE
SPACE BY FINDING THE NODE SETS OF ALL CONNECTED
COMPONENTS OF G.
     :
************
SEQUENCE CHART
************

     :
LOOP TO PROCESS COMPONENTS
*
*    INITIALIZE SETS FOR ARCS AND NODES GET FIRST NODE
*    FROM THE LEFT-OVER.
*         :
*         :
*         :
*    LOOP TO GET ADJACENT NODES IN T
*    *
*    *
*    *
*    *
*    END OF LOOP, NODES OF ONE COMPONENT ARE IN T, TAKE
*    THEM OUT FROM N.
*         :
*         :
END OF LOOP
     :
     :
```

```
      SUBROUTINE COCYCL(G,C)



      GRAPH G
      SET STAQUE C



      IMPLICIT SET(A-Z)



      N = NODES(G)

      DO 20 WHILE (N,NE,&)


      A = &
      T = &
      S = ELT(1,N)

      DO 10 WHILE (S,NE,&)
      T = T,UN,S
      A = STAR(G,S),DF,A
   10 S = INC(G,A),DF,T


      C = C : T
   20 N = N,DF,T

      RETURN
      END
```

```
 *****
 TITLE
 *****
SPANNING TREE
     :
     :
 *************
 SPECIFICATION
 *************
THIS SUBROUTINE GENERATES A DIRECTED SPANNING TREE
(GRAPH) WITH ROOT U FOR THE CONNECTED COMPONENT OF
G CONTAINING THE NODE U.
     :
 ***************
 SEQUENCE CHART
 ***************
INITIALIZE TREE AND NODE SET S WITH U,
ARC SET T  WITH COBOUNDARY OF U.
     :
     :
     :
LOOP TO PROCESS OUTGOING ARCS FROM NODE SET S
*
*    LOOP TO PROCESS ARCS
*    *
*    *    GET END NODE WHICH IS NOT IN S
*    *         :
*    *         :
*    *    AND ADD IT TO S AND ASSIGN IT TO T
*    *    *
*    *    *
*    *    *
*    *    *
*    END OF LOOP GET NEW SET OF OUTGOING ARCS
*         :
END OF LOOP
     :
     :
```

```
      SUBROUTINE SPTREE(G,U,TREE)


      GRAPH G, TREE
      SET U



      IMPLICIT SET(A-Z)



      ASSIGN TREE, U
      S = U
      T = COB(G,U)

      DO 20 WHILE (T.NE.&)

      DO 10 FORALL A.IN.T

      W = BD(G,A)
      Y = W.DF.S

      IF (Y.EQ.&) GO TO 10
      S = S.UN.Y
      X = W.DF.Y
      ASSIGN TREE, X - Y, A
   10 CONTINUE

   20 T = COB(G,S)

      RETURN
      END
```

```
 *****
 TITLE
 *****
FUNDAMENTAL CYCLES
     :
     :
 *************
 SPECIFICATION
 *************
TREE IS ASSUMED TO BE A DIRECTED SPANNING TREE OF
ONE OF THE COMPONENTS OF G. FROM THIS SPANNING TREE
THIS SUBROUTINE GENERATES, IN A STANDARD MANNER, A
BASIS FOR THE CYCLE SPACE OF THE PARTICULAR COMPONENT.
     :
 ***************
 SEQUENCE CHART
 ***************
GET ARCS OF G WHICH ARE NOT IN TREE
     :
LOOP TO GET THE CORRESPONDING CYCLE OF THESE ARCS
*
*
*
*    IF THIS IS A SELF-LOOP,THEN THIS IS A CYCLE
*    *
*    *... NOT A SELF-LOOP
*    *    GO BACK TO THE ROOT OF THE TREE FROM BOTH END
*    *    NODES OF THE ARC IN A LOOP
*    *    *
*    *    *
*    *    *    QUIT LOOP IF THE PARALLEL PROCESS OF THE
*    *    *    TWO PATHS MET AT THE BRANCH-ROOT
*    *    *         :
*    *    *    COLLECT THE ARCS IN S
*    *    *         :
*    *    *         :
*    *    END OF LOOP
*    COLLECT CYCLE
*         :
END OF LOOP
     :
     :
```

```
      SUBROUTINE FNDCYC(G,TREE,CYCLES)


      GRAPH G,TREE
      SET STAQUE CYCLES




      IMPLICIT SET(A-Z)



      X = STAR(G,NODES(TREE)).DF.ARCS(TREE)

      DO 20 FORALL A.IN.X
      S = A
      T = INC(G,A)

      IF (SIZE(T).EQ.1) GO TO 20



      DO 10 WHILE (T.NE.&)
      T = NCOB(TREE,T)


      IF (T.EQ.&) GO TO 20

      S = S.SM.T
   10 T = PBD(TREE,T)

   20 CYCLES = CYCLES : S   .

      RETURN
      END
```

```
 *****
 TITLE
 *****
FUNDAMENTAL CUTS
      :
      :
 *************
 SPECIFICATION
 *************
TREE IS ASSUMED TO BE A DIRECTED SPANNING TREE OF
A COMPONENT OF G, AND FROM TREE THIS SUBROUTINE
GENERATES IN THE STANDARD MANNER A BASIS OF THE
COBOUNDARY SPACE OF THE,COMPONENT.
      :
 **************
 SEQUENCE CHART
 **************

LOOP FOR ALL ARCS IN THE TREE
*
*
*
*     LOOP TO GET ALL NODES OF THE SUBTREE INTO S
*     *
*     *
*     *
*     *
*     END OF LOOP GET FUNDAMENTAL CUT FROM THE NODES IN S
*           :
END OF LOOP
      :
      :
```

```
      SUBROUTINE FNDCUT(G,TREE,CUTS)


      GRAPH G, TREE
      SET STAGUE CUTS



      IMPLICIT SET(A-Z)


      DO 20 FORALL A,IN,ARCS(TREE)
      S = &
      T = A

      DO 10 WHILE (T,NE,&)
      R = NBD(TREE,T)
      S = S,UN,R
   10 T = PCOB(TREE,R)

   20 CUTS = CUTS : COB(G,S)

      RETURN
      END
```

```
 *****
 TITLE
 *****
SHORTEST PATH.
      :
      :
      :
      :
      :
 *************
 SPECIFICATION
 *************
G IS A DIRECTED GRAPH IN WHICH EACH ARC HAS A GIVEN
NONNEGATIVE LENGTH, THE SUBROUTINE FINDS A SHORTEST
PATH FROM NODE START TO NODE TERM, AND RETURNS IT IN
THE LIST PATH, IF NO SUCH PATH EXISTS, THE LIST WILL
BE EMPTY, THE REAL NUMBER INF REPRESENTS INFINITY, IT
IS ASSUMED TO BE LARGER THAN THE SUM OF THE LENGTH OF
ALL ARCS OF G. THE LENGTH OF THE FINAL PATH WILL BE
IN M, AND THIS NUMBER WILL BE EQUAL TO INF, IF NO
PATH EXISTS.
 ******
 METHOD
 ******
THE METHOD USED IS THE BIDIRECTIONAL SEARCH BY
I.POHL: BI-DIRECTIONAL AND HEURISTIC SEARCH IN
        PATH PROBLEMS. TECHN,REPORT CS-136,
        STANFORD UNIVERSITY, 1969.
 **************
 DATA-STRUCTURE
 **************

NODES REACHED FROM START AND TERM, RESPECTIVELY
      :
NODES NOT REACHED BUT REACHABLE ALONG ONE ARC FROM
THE NODES REACHED FROM START AND TERM, RESPECTIVELY
      :
CURRENT DISTANCE OF A REACHED NODE FROM START OR
TERM, RESPECTIVELY
      :
CURRENT ARC LEADDING FROM A NODE OR TO A NODE WHICH
WAS REACHED
      :
MINIMAL DISTANCE TO THE NODES REACHABLE ALONG ONE ARC
FROM START AND TERM, RESPECTIVEL
      :
FLAG TO SET WHEN THE TWO DIRECTIONAL SEARCH MEET
      :
TEMPORARY VARIABLES
```

```
SUBROUTINE SHPATH(G,START,TERM,LENGTH,INF,M,PATH)


GRAPH G
SET START, TERM
REAL PROPERTY LENGTH
REAL INF, M
SET STAQUE PATH


SET  S , T

SET  SR , TR

REAL PROPERTY  SDIST°, TDIST

SET PROPERTY  IN , OUT

REAL SMIN, TMIN

LOGICAL  FLAG
```

```
     :
     :
 **************
 SEQUENCE CHART
 **************
INITIALIZATION
     :
     :
     :
     :
     :
     :
     :
     :
     :
     :
     A FICTITIOUS ARC w IS INSERTED WITH LENGTH INF TO
     ENSURE THAT THERE IS AT LEAST ONE PATH
          :
          :
          :
LOOP FOR BIDIRECTIONAL SEARCH
*
*    CHECK WHETHER PATH CAN BE FOUND
*    *
*    *... NO, CLEAR PATH AND GO TO EXIT
*    *          :
*    *          :
*    YES, TAKE NEXT STEP
*    *
*    *
*    *
*    *... STEP FROM START
*    *          COLLECT NODES WITH MINIMAL DISTANCE FROM
*    *          THE NODES REACHABLE ALONG ONE ARC
*    *               :
*    *               :
*    *               :
*    *               :
*    *               :
*    *               :
*    *               :
*    *               :
*    *               :
*    *               :
*    *          ADD COLLECTED NODES TO THE SET OF REACHED
*    *          NODES, SUBTRACT THEM FROM THE REACHABLE
*    *          NODES, AND FIND NODES WHICH ARE REACHABLE
*    *          FROM THEM ALONG ONE ARC.
*    *               :
```

```
      SET  W,X,Y,Z,U
      REAL A,B



      SDIST(START) = 0.
      TDIST(TERM) = 0.
      S = START
      SR = S
      T = TERM
      TR = T
      SMIN = 0.
      TMIN = 0.
      FLAG = .FALSE.
      M = 0.


      W = CREATE(0)
      LENGTH(W) = INF
      ASSIGN G, START - TERM, W

      DO 130 WHILE (.NOT.FLAG)

      IF (M.LT.INF) GO TO 10

      PATH = #
      GO TO 170

10    M = INF
      PATH = #
      IF (SMIN.GT.TMIN) GO TO 70



      DO 40 FORALL X,IN,SR
      A = INF
      IF (CHECK(SDIST,X)) A = SDIST(X)
      IF (A-M) 20,30,40
20    M = A
      PATH = X
      GO TO 40
30    PATH = X : PATH
40    CONTINUE
      SMIN = M



      DO 60 WHILE (PATH.NE.#)
```

```
*    *              :                                        X = DFIRST(PATH)
*    *              MEANWHILE CHECK IF OTHER SIDE WAS
*    *              REACHED
*    *              *                                        IF (.NOT.(X,IN,T)) GO TO 50
*    *              *... YES, SET FLAG, SAVE NODE
*    *              *         :                              FLAG = .TRUE.
*    *              *         :                              U = X
*    *              *... TRANSFER NODE TO THE REACHED SET
*    *                        :                        50    SR = SR.DF.X
*    *                        :                              S = S.UN.X
*    *                   LOOP TO GET REACHABLE NODES FROM
*    *                   THIS NODE
*    *                   *                                  DO 60 FORALL Z.IN.PCOB(G,X)
*    *                   *                                  Y = NBD(G,Z)
*    *                   *                                  B = M+LENGTH(Z)
*    *                   *                                  A = INF
*    *                   *                                  IF (CHECK(SDIST,Y)) A = SDIST(Y)
*    *                   *                                  IF (A.LE.B) GO TO 60
*    *                   *                                  SDIST(Y) = B
*    *                   *                                  IN(Y) = Z
*    *                   *                                  SR = SR.UN.Y
*    *                   *                        60        CONTINUE
*    *                   END OF LOOP
*    *                   :                                  GO TO 130
*    *         STEP FINISHED, GO FOR NEXT STEP
*    *... STEP FROM TERM
*              COLLECT NODES WITH MINIMAL DISTANCE FROM
*              THE NODES REACHABLE ALONG ONE ARC
*                   :                                  70    DO 100 FORALL X.IN.TR
*                   :                                        A = INF
*                   :                                        IF (CHECK(TDIST,X)) A = TDIST(X)
*                   :                                        IF (A-M) 80,90,100
*                   :                                  80    M = A
*                   :                                        PATH = X
*                   :                                        GO TO 100
*                   :                                  90    PATH = X : PATH
*                   :                                  100   CONTINUE
*                   :                                        TMIN = M
*              ADD COLLECTED NODES TO THE SET OF REACHED
*              NODES, SUBTRACT THEM FROM THE REACHABLE
*              NODES, AND FIND NODES WHICH ARE REACHABLE
*              FROM THEM ALONG ONE ARC
*                   :                                  DO 120 WHILE (PATH.NE.#)
*                   :                                  X = DFIRST(PATH)
*                   MEANWHILE CHECK IF OTHERSIDE WAS
*                   REACHED
*                   *                                  IF (.NOT.(X,IN,S)) GO TO 110
*                   *... YES, SET FLAG, SAVE NODE
*                   *         :                        FLAG = .TRUE.
*                   *         :                        U = X
```

```
*     *              :                                          X = DFIRST(PATH)
*     *              MEANWHILE CHECK IF OTHER SIDE WAS
*     *              REACHED
*     *              *
*     *              *... YES, SET FLAG, SAVE NODE              IF (.NOT.(X.IN.T)) GO TO 50
*     *              *         :                                FLAG = .TRUE.
*     *              *         :                                U = X
*     *              *... TRANSFER NODE TO THE REACHED SET
*     *                        :                          50    SR = SR.DF.X
*     *                        :                                S = S.UN.X
*     *                   LOOP TO GET REACHABLE NODES FROM
*     *                   THIS NODE
*     *                   *                                    DO 60 FORALL Z.IN.PCOB(G,X)
*     *                   *                                    Y = NBD(G,Z)
*     *                   *                                    B = M+LENGTH(Z)
*     *                   *                                    A = INF
*     *                   *                                    IF (CHECK(SDIST,Y)) A = SDIST(Y)
*     *                   *                                    IF (A.LE.B) GO TO 60
*     *                   *                                    SDIST(Y) = B
*     *                   *                                    IN(Y) = Z
*     *                   *                                    SR = SR.UN.Y
*     *                   *                          60        CONTINUE
*     *                   END OF LOOP
*     *                        :                               GO TO 130
*     *         STEP FINISHED, GO FOR NEXT STEP
*     *... STEP FROM TERM
*          COLLECT NODES WITH MINIMAL DISTANCE FROM
*          THE NODES REACHABLE ALONG ONE ARC
*               :                                     70      DO 100 FORALL X.IN.TR
*               :                                             A = INF
*               :                                             IF (CHECK(TDIST,X)) A = TDIST(X)
*               :                                             IF (A-M) 80,90,100
*               :                                     80      M = A
*               :                                             PATH = X
*               :                                             GO TO 100
*               :                                     90      PATH = X : PATH
*               :                                     100     CONTINUE
*               :                                             TMIN = M
*          ADD COLLECTED NODES TO THE SET OF REACHED
*          NODES, SUBTRACT THEM FROM THE REACHABLE
*          NODES, AND FIND NODES WHICH ARE REACHABLE
*          FROM THEM ALONG ONE ARC
*               :                                             DO 120 WHILE (PATH.NE.#)
*               :                                             X = DFIRST(PATH)
*               MEANWHILE CHECK IF OTHERSIDE WAS
*               REACHED
*               *                                             IF (.NOT.(X.IN.S)) GO TO 110
*               *... YES, SET FLAG, SAVE NODE
*               *         :                                   FLAG = .TRUE.
*               *         :                                   U = X
```

17. APPENDIX. EVALUATION OF EXPRESSIONS

THE FGRAAL COMPILER WAS CONSTRUCTED BY AUGMENTING THE EXISTING RALPH COMPILER FOR THE UNIVAC 1108.

THE RALPH COMPILER ASSUMES THAT NO FUNCTION CAN MODIFY ITSELF AND THUS TAKES CERTAIN LIBERTIES WITH THE EVALUATION OF EXPRESSIONS AND ELEMENTS IN AN ARGUMENT LIST. AN EXPRESSION OR ARGUMENT LIST IS SCANNED RIGHT TO LEFT AND EACH TIME A FUNCTION IS ENCOUNTERED, IT IS EVALUATED, EACH TIME AN OPERATOR IS ENCOUNTERED WHICH HAS STRICTLY GREATER PRECEDENCE THAN THE OPERATOR TO ITS LEFT, IT IS EVALUATED, AND PROCESSING CONTINUES BY EXAMINING THE PRECEDENCE RELATION BETWEEN THE OPERATOR TO THE RIGHT OF THE NEWLY EVALUATED RESULT AND LOOKING FOR A STRICTLY GREATER PRECEDENCE THAN THE OPERATOR TO ITS LEFT. THUS THE EXPRESSION

F(A) + B * C * D + F(E)

WILL BE EVALUATED AS THOUGH IT WERE PARENTHESIZED AS FOLLOWS

$$(((F(A))_4 + ((B * C)_2 * D)_3)_5 + (F(E))_1)_6$$

WHERE THE NUMBER AT THE BASE OF THE RIGHT PARENTHESIS GIVES THE ORDER IN WHICH THE EVALUATIONS ARE MADE. THIS PARSING ALGORITHM YIELDS PERFECTLY VALID RESULTS AS LONG AS FUNCTIONS ARE NOT PERMITTED TO MODIFY THEIR ARGUMENTS OR SOME GLOBAL VARIABLES, I.E. SO LONG THERE ARE NO SIDE EFFECTS.

HOWEVER SIDE EFFECTS DO EXIST IN GRAAL. FOR EXAMPLE, THERE ARE OPERATORS (FUNCTIONS) THAT ARE ALLOWED TO MODIFY THEIR ARGUMENTS, E.G. DFIRST(L). BECAUSE OF THIS, IF A VARIABLE IS USED MORE THAN ONCE IN THE CONTEXT OF AN EXPRESSION OR ARGUMENT LIST, AND AT LEAST ONE SUCH USE IS AS AN ARGUMENT OF A FUNCTION OR OPERATOR WHICH MODIFIES IT, THE UNUSUAL ORDER OF EVALUATION USED BY THE RALPH COMPILER MUST TAKE INTO CONSIDERATION IN ORDER TO SIMULATE THE ACTUAL RESULTS.

EXAMPLE 1. LET

```
L = 1 : 2 : 3 : 4 : 5
```

THEN THE STATEMENT

```
X = DFIRST(L) - DFIRST(L)
```

IS EQUIVALENT TO

```
X = 2 - 1
```

AND NOT X = 1 - 2, SINCE THE RIGHTMOST DFIRST OPERATOR IS EVALUATED FIRST.

EXAMPLE 2. THE FUNCTION CALL

```
FUNC(DFIRST(L),L,DFIRST(L))
```

WITH THE PREVIOUSLY DEFINED L, IS EQUIVALENT TO

```
FUNC(2,3:4:5,1)
```

SINCE THE RIGHTMOST DFIRST IS EVALUATED FIRST,
THE SECOND ARGUMENT IS NOT EVALUATED SINCE
THERE IS NO OPERATOR OR FUNCTION IN IT,
AND THE LEFTMOST DFIRST IS EVALUATED SECOND,
WHICH LEAVES LEAVES L WITHOUT ITS ORIGINAL
FIRST TWO ELEMENTS.

EXAMPLE 3. EVEN PARENTHESES DO NOT HELP

```
X = (DFIRST(L)-DFIRST(L))*DFIRST(L)
```

IS EQUIVALENT TO

```
X = (3-2)*1
```

ONE WAY TO AVOID THIS PROBLEM WOULD BE TO BREAK THE EXPRESSION DOWN INTO SEPARATE STATEMENTS. THUS WE CAN REWRITE EXAMPLE 1. AS

```
X1 = DFIRST(L)
X = X1 - DFIRST(L)          (1-2)
OR  X = DFIRST(L) - X1      (2-1)
```

DEPANDING UPON WHICH EXPRESSION WE ACTUALLY WANT. WE CAN REWRITE EXAMPLE 2 AS

```
M = L
X = DFIRST(L)
   ... FUNC(X,M,DFIRST(L))
```

TO YIELD

```
   ... FUNC(1,1:2:3:4:5,2)
```

18. REFERENCES

RHEINBOLDT, W.C., BASILI, V.R. AND MESZTENYI, C.K. [1971].
ON A PROGRAMMING LANGUAGE FOR GRAPH ALGORITHMS,
UNIVERSITY OF MARYLAND, COMPUTER SCIENCE CENTER,
TECHNICAL REPORT TR - 158. SUBMITTED FOR PUBL. IN BIT.

- - - , [1972],
GRAAL - A GRAPH ALGORITHMIC LANGUAGE,
PROCEEDINGS OF THE SECOND IBM RESEARCH SYMPOSIUM
ON SPARSE MATRICES AND THEIR APPLICATIONS,
SEP. 9-10, 1971. PLENUM PRESS, N.Y.

UNCLASSIFIED
Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)*

University of Maryland
Computer Science Center
College Park, Maryland 20742

2a. REPORT SECURITY CLASSIFICATION

Unclassified

2b. GROUP

3. REPORT TITLE

FGRAAL, Fortran-extended Graph Algorithmic Language

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*

Technical Report, March 1972

5. AUTHOR(S) *(First name, middle initial, last name)*

Victor R. Basili
Charles K. Mesztenyi
Werner C. Rheinboldt

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
March 1972	51	2

8a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)
N00014-67-A-0239-0021 b. PROJECT NO. NGL-21-002-008 NR 044-431	Technical Report TR-179
c.	9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*
d.	

10. DISTRIBUTION STATEMENT

Approved for public release;
distribution unlimited

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
	Mathematics Program Office of Naval Research Arlington, Virginia 22217

13. ABSTRACT

This report describes the Fortran version FGRAAL of the graph algorithmic language GRAAL (Technical Report TR-158) as it has been implemented for the Univac 1108. FGRAAL is an extension of FORTRAN V and is intended for describing and implementing graph algorithms of the type primarily arising in applications. The formal description contained in this report represents a supplement to the FORTRAN V manual for the Univac 1108 (UP-4060), that is, only the new features of the language are described. Several typical graph algorithms, written in FGRAAL, are included to illustrate various features of the language and to show its applicability.

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
GRAPH LANGUAGE						
FORTRAN V EXTENSION						
GRAPH ALGORITHMS						