(7)

FEASIBILITY STUDY FOR THE APPLICATION OF PMSL: A NOTATION AND DATA BASE TO DESCRIBE AND ANALYZE COMPUTER SYSTEMS

Michael Knudsen
Gordon Bell
Allen Newell

May 30, 1972

Department of Computer Science
Carnegie-Mellon University
Pittsburgh, Pennsylvania

ABSTRACT

This study is concerned with applying (and extending) the PMS notation (Bell and Newell, 1971). Knudsen's work on a computer system, PMSL, which implements, extends and disambiguates the PMS notation provides the basis for this study. The PMSL system provides for: PMS to be used in the description, input and output of computer systems; machine structures to be accessed and compared; various attributes to be calculated and accessed; and reliability and performance calculations to be carried out in terms of PMS attributes.

The purpose of this study is to explore the feasibility of using the PMS system as a data base and analysis system for a set of computer system descriptions. Such a system would allow computer systems to be specified to any level of detail, questions to be answered about individual computers, comparative studies to be made among computer systems, and inventories and census to be built of classes and families of systems.

FEASIBILITY STUDY FOR THE APPLICATION OF PMSL

PMS is a notational scheme for describing computer systems at the highest structural level, in terms of processors, switches, memories, etc., with the information transactions between these components characterized in gross quantitative terms (essentially information units and information-units per unit time). PMS was developed by Bell and Newell in attempting to describe a large number of computer structures (Bell and Newell, 1971). A brief description of the language is given in Bell and Newell (1970, copy attached); and a brief account of its origin is given in (Bell and Newell, 1969, copy attached). Although it is still somewhat early to tell, there are some indications (e.g., see Brooks, 1971) that the notation provides an adequate tool for the high level description of computer systems, essentially at the level of functional and performance specifications, descriptions for manuals, and descriptions for inventories and census. The level of detail of the notation is variable and can be extended downward to any desired degree, all the way to registers, adders and logic gates.* The notation does not describe the programming level of computer systems, e.g., instruction sets, operating systems, language systems, etc. A notation for describing instruction sets, called ISP, was developed concurrently with PMS, but the present proposal does not extend to descriptions at the program level.

---

* The PMS notation is being used, in fact, as the basis for a language of register transfer modules (called RTM's and available commercially as the PDP16; see Bell, Eggert, Grason and Williams, 1972). The application is not directly relevant to the present contract.

The original notation was essentially a publication language, built for human use. Knudsen has developed a programming system, called PMSL (for PMS language), which is an implementation and extension of PMS in operational form. It is written in SNOBOL and operates interactively on the PDP10. It permits specifying computer configurations in PMS notation to the system, with consequent printing and display of such structures. An important feature of PMS is its relative freedom in the use of notation and abbreviation (essentially, anything that is meaningful and unambiguous in the context of the existing PMS structure can be stated). This has been preserved in Knudsen's system. The system holds a data-base of specifications, some representing specific systems, and retrieves specific items of data from descriptions, which includes determining whether a presented configuration is a valid instance of any of the classes of configurations held in the data base. Since information is available by class, this involves non-trivial search and match operations. Simple functions include computing costs and other aggregate functions such as memory size. Quite complex calculations include the reliabilities (given a minimum criterion) and performances of configurations. Others can also be included. These involve approximations, of course, since what is available is only a high-level description, and the exact performance has to be based on job mix, etc.

The above capabilities exist at various levels of completeness, sophistication and efficiency. The work is Knudsen's Ph.D. thesis. The thesis draft is being prepared now, and should be completed in late June. A short annotated example of the output of PMSL is attached, which is understandable given a knowledge of PMS.

Interest to NBS

The basic value of such a system to NBS lies in its ability: (1) to specify configurations in a relatively uniform way; (2) to update such descriptions continuously and rapidly; (3) to process such descriptions for completeness, consistency and accuracy in various ways; (4) to determine various global properties of such configurations; (5) and to provide displays and printed records of such configurations that are known to be error-free and up to date. The uses of such a capability range from a device for system specification in ongoing procurement, to an automated catalogue of available computers, to a system to study the impact of various standards and pricing policies, to an archive of all existing computers.

General Activities Proposed

The major goal of this contract is to determine whether the vehicle we now have (or one likely to be developed from it) is really suitable for the uses described in the previous section: This involves four things:

1. The determination of the realistic dimensions of such uses, e.g., the sizes of realistic configurations described to the level of detail actually necessary, the sizes of the data bases that would be required for various realistic application, etc. This determination should not be just in terms of general characteristics, but should include a small set of actual examples of potential use. This would entail encoding several real systems. We intend to take all the models of the IBM System/360 and 370 series, and also one machine from another manufacturer, e.g., the DEC PDP-10. Alternatively, a catalogue of minicomputers would provide a very nice data base study.

2. The determination of the performance of the PMSL system along such dimensions. This would comprise both experiments with the system and

analyses of the essential nature of the computation for extrapolation further out on along the dimensions (e.g., whether the reliability calculation method is feasible for systems of realistic size). This would undoubtedly require further development of the system in an attempt to overcome the deficiencies of the present scheme (which are sure to be there). It would seem necessary to run a set of accumulated examples, including even building up a moderate sized data base, to obtain a genuine feeling for the usefulness of the system.

3. Additional work on the human engineering of the user interface, including the layout of the printed and plotted outputs. Though minor in some respects, the capabilities of the system must be kept excellent enough along this dimension to present an unclouded picture of how the system might operate given further polishing. In this later regard, we have a very interesting printing device, the Xerox LDX printer, which has a 200 point/inch resolution and is capable of drawing the PMS diagrams we use. This might be incorporated as the catalogue output medium.

4. The determination of the computing vehicle for using such a system in practice. The demands vary, of course, with the particular application under consideration, and the solutions could correspondingly vary from a single system at NBS, to a centralized system accessible by network, to a set of regional systems, to making versions of the system available to all participants in a given type of activity (e.g., all vendors interested in bidding on a class of large computer systems). The main concern here is to find out whether (1) the present SNOBOL implementation is basically sound and can be made suitable with only minor polishing; (2) a single new system should be built, or (3) systems must be built in many different languages for many different machines. With regard to the latter, an important problem is to see whether it is even possible to implement realistic versions in widely available

languages (essentially FORTRAN and COBOL) operating in batch mode. The present version in SNOBOL* is interactive and sophisticated in its use of dynamic data structures and these appear to be essential to an appropriate system.

The present study would include detailed exploration of the above issues and the proposal of possible solutions, including testing of software schemes to be sure the proposed solutions were feasible. It could include a rather complete design for re-implementation, if that should prove desirable. However, development of a new production-oriented version in a new language is not possible within a limited study contract such as this.

We have interests in developing more advanced capabilities, such as finding bottlenecks in a computer system, balancing a system, etc. However, we take the view that it is the basic capabilities that are important in application. Thus, in this contract we will place the major effort on exploring what we have now in PMSL in application to a real environment, rather than advancing the PMSL's capabilities. There will be exceptions to this, but they will be dictated by the demands of particular applications.

Specific, Proposed Activities

The above general categories fall into several specific activities. These are arranged roughly by time, although certain activites like liaison and documentation are continous.

1. Data base creation and system testing

We will experiment with creating a data base containing detailed descriptions of a few selected computers, and subject these to all existing and future analyses and operations. In so doing, we expect the following results:

---

* SNOBOL was originally selected because it is widely available and has the necessary capabilities.

a. In the initial creation of the data bases, more will be learned about the real day-to-day issues and problems of using PMSL. This work will suggest improved human-engineering features, particularly those dealing with secondary memory. Actually, incremental improvements in user convenience will occur continually, but the data base is a major effort.

b. During creation and use, we will no doubt examine and change current conventions, and extend and modify these accordingly.

c. The behavior of certain of our algorithms when confronted with large and complex structures will be examined. In particular, the reliability-estimating function is combinatorial in nature and has yet to be run on a large structure which might slow its operation so much as to require its redesign; on the other hand, its performance may well be acceptable as it now stands. Testing is necessary.

d. The sample data bases will be a convincing demonstration of PMSL, and a part of its documentation.

2. Dimensional Computing

The concept of a numeric value will be extended to include dimensioned quantities. That is, arithmetic and other operations will deal with numbers with or without units, such that in any context where a number is valid, so is "5 watt/sec " or "36 bit/word." Addition would require equivalent dimensions and make required conversions; multiplication would compute the appropriate dimensions of the result. Dimensioned quantities are already widely used and recognized in PMSL, but presently no arithmetic operations on them are possible, except by ignoring the given units and assuming others.

3. Iteration and Deferred Execution of Commands

Currently PMSL executes individual command statements as they are typed in and then discards them, although their results (side effects) may remain. There is no way to iterate a statement or group of sequenced statements over a list or set of similar data-objects. In particular, there is not way to process such a set to obtain a result-value which is a property of the set as a whole (e.g., the sum of a list of numbers), except for some highly specific PMSL system functions. This need not involve going to a full programming language with user-defined functions and labels and goto's, although some of these capabilities may be required.

4. Configuration Classes

We give high priority to extending the PMS notation to represent families of computer-system configurations. The present system handles context free classes of systems, where the possibilities for each component in a system are independent of the selections for other components. It also handles several cases of dependence between components, e.g., the word size of a disk being specified as the word size of the primary memory. However, as one moves toward more complex dependencies the present system becomes cumbersome, and additional notational forms may be required. The types of dependencies that arise in practice can be illustrated by the configuration diagram of the IBM 1800 (see pages 401-403 in Bell and Newell, 1971).

Testing whether a given structure satisfies a given specification (i.e., is a member of a specified family of systems) is a central operation in the PMSL system. Currently we can do this for the simple families of systems that are now expressible in the system. This algorithm will be extended to the full range of the current language. Furthermore, any increase in the

complexity of the dependencies expressible in the language will have to be incorporated in the algorithm.

5. Generalized Graph Plotting

The present interactive plotting facility (which uses the terminal or line printer and eventually the Xerox LDX system) will be extended to allow any user-defined PMSL function to be plotted. Currently plotting is restricted to a fixed set of Processor and Memory parameters, performance and cost results from a fixed set of built-in functions, constants, and basic arithmetic operations on these. Within this domain it is quite general but cannot, for example, involve reliability estimates for components.

6. Documentation

We will produce the following documentation:

a. A user's manual (parts of which already exist), which will be updated as required.

b. A description of the structure of PMSL and its algorithms. This will be in sufficient detail to allow a good systems programmer to build a PMSL system.

c. A description of details of our implementation, to permit others to understand and modify our system.

d. A report to be made before the end of the first year that describes the set of applications to be considered, the data bases to be used, the initial view on ultimate computer implementations, etc. This will serve as a progress report on the first year's work, though it will be produced as soon as the details of the study become clear enough.

e. A final report (which may be in several distinct parts) that gives the results of all the studies and the overall conclusions that have been reached. Documentation on the details of the data bases, etc., will also be provided.

7. Liaison with NBS

For the system to be effective, it will be important for someone at NBS to be involved with it at least in a monitoring capacity. This will not only provide the contractor with better feedback about the real progress, but it will allow us to obtain information about how such a system might be used in the activities of the bureau.

Other Work

There have been several types of activities that bear some relation to PMS and the proposed investigation: Weik's census on all computers (four versions, up through 1964); Knight's comparative study of computer technology (Knight, 1966); and various commercial continuing census (Keydata, Auerback Associates, Datapro). The present effort appears to us to go far beyond these in the direction of the application described above, in the degree of sophistication and completeness attempted, and in the development of an operational computer system. Thus, we do not present a detailed comparison of the proposed research with these other efforts.

Facilities

The work will be done in the Computer Science Department of Carnegie-Mellon University, which is where both the original work on PMS and Knudsen's thesis have occurred. There is a well equipped computational facility

consisting essentially of two PDP-10 systems which operate in time sharing mode, with an assortment of terminals, and secondary memory. In addition, there are several PDP-11's, now being put together into a multi-processor. This should be operational within the period of the contract, though its impact on the proposed research will probably not be great.

The environment itself supports a wide range of computer science research, from work on computer structures, to implementation languages, to artificial intelligence. Thus there will be adequate conceptual support for the study. Two recent Annual Reviews of Research of the Computer Science Department are included as supporting material in order to give some indication of this environment.

Budget

The main effort on this contract will be that of Mike Knudsen as a post-doctoral research associate. His salary, together with overhead and fringe benefits,are to be paid from the contract. Gordon Bell and Allen Newell will be co-principal investigators and will remain active in both a supervisory and a consultative role. No costs are included in the contract for them. Support is included for one research assistant, beginning in January of the first year, to assist in encoding the various machines and in evaluating the system's performance. Some travel is requested, primarily for Knudsen to visit NBS. This is essential in order that we may obtain a view of the NBS activities thereby shaping our own work and priorities. No funds are included for computation, since we believe that we can absorb these costs into our ongoing research program. This will imply, in effect, that

a fraction of this effort will be supported by our research contract with the Advanced Research Projects Agency (F44620-70-C0107) monitored by the Air Force Office of Scientific Research.

The contract is written for two years. The only change in the second year being the 5.5% increase in salary for Knudsen.

NATIONAL BUREAU OF STANDARDS

Feasibility Study for the Application of PMSL

PROPOSED BUDGET

	7/ 1/72 6/30/73	7/ 1/73 6/30/74
Post Doctoral Fellow	$15,000	$15,825
Graduate Assistant	2,260	4,520
Fringe Benefits @ 12%	180	190
Travel	500	500
Total direct costs	$17,940	$21,035
O/H at 46% of S & W	6,900	7,280
GRAND TOTAL	$24,840	$28,315

References

1. C. G. Bell, J. L. Eggert, J. Grason, and P. Williams, "The description and use of Register-Transfer-Modules (RTM's)," IEEE Transactions on Computers, vol. C-21, May 1972, 495-500.

2. C. G. Bell and A. Newell, "PMS and ISP," Computer Science Research Review, Computer Science Department, Carnegie-Mellon University, 1969.

3. C. G. Bell and A. Newell, "The PMS and ISP descriptive systems for computer structures," AFIPS Conference Proceedings, Spring Joint Computer Conference, 1970, 351-354.

4. C. G. Bell and A. Newell, Computer Structures: Readings and Examples, McGraw-Hill, 1971.

5. F. P. Brooks, Jr., "Review of Bell, C. G. and Newell, A., Computer Structures: Readings and Examples," Computing Reviews, vol. 12, May 1971, 245-248.

6. K. E. Knight, "Changes in computer performance," DATAMATION, vol. 12, September 1966, 40-54.

7. M. H. Weik, A Fourth Survey of Domestic Electronic Digital Computer Systems, Report 1227, Ballistic Research Laboratories, Aberdeen, Md. January, 1964. (See also earlier versions in 1961 and 1955.)

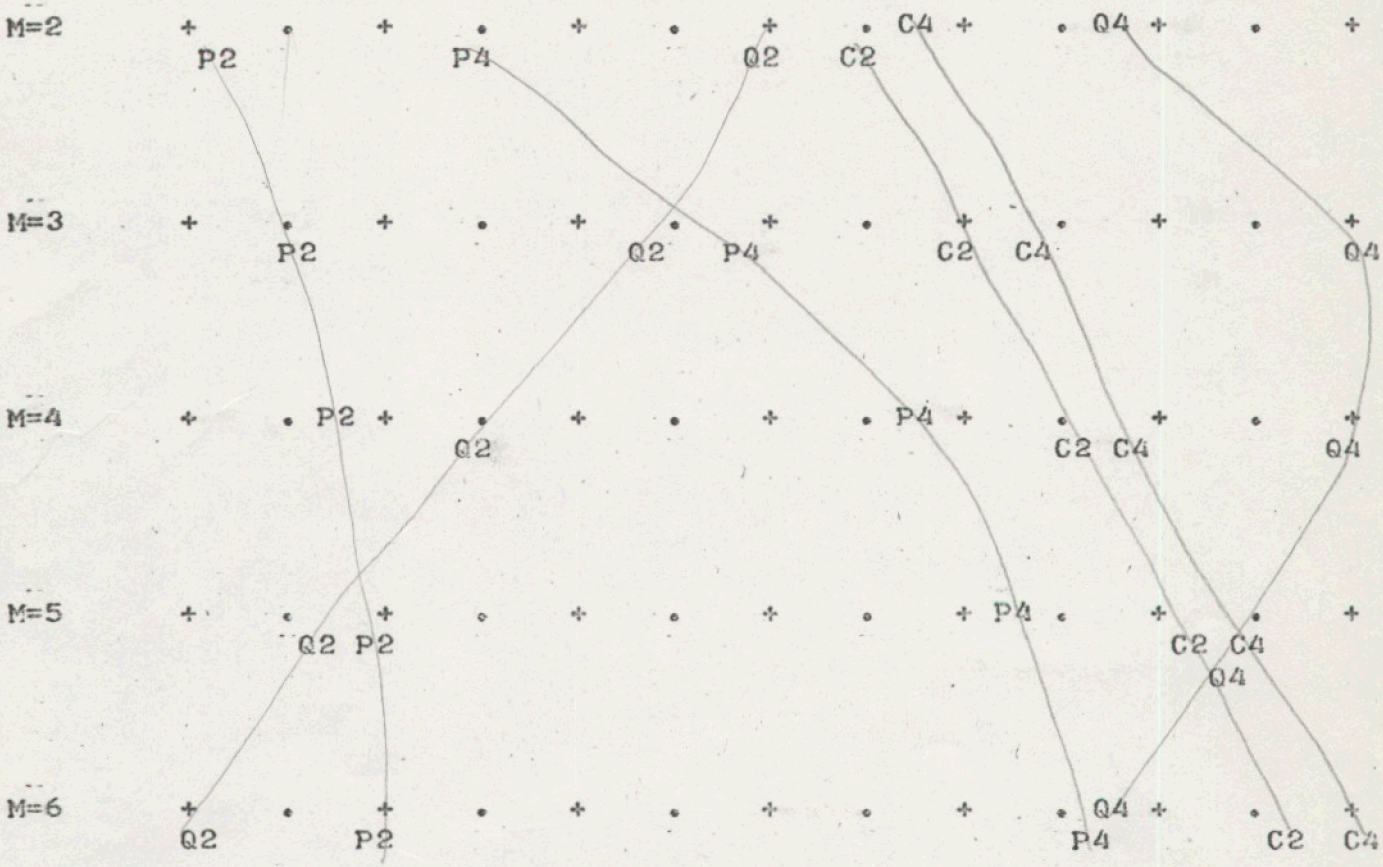
```
.R SNOBOL 70
*PMSL1
PMSL1 17-FEB-72 FULL USE OF INDEF EXPRS IN VALUE-CHECKING AND IMPATT;
 NEW XREP; INSPECT CAN TYPE BLURBS.  CONCAT ADDED 5-APR.
NOW USING TABS IN SOURCE FILES.
29-MAY-72     14:56.34
←* DEFINE  BASIC TYPES
←COMPONENT//CO:=''FE(REL//RELIABILITY:<Ø.Ø--1.Ø>;COST:<[+INTEGER]>)
>>>LOADING INDFEXEVAL:INDFEX.LIB/
INDFEXEVAL,MULTOUT,IXPATS,
FE
←* ABBREVIATIONS CAN HAVE THE SHORT NAME FIRST--

←M//MEM//MEMORY::=''CO(SIZE:<[+INT] <W//WORD!BYTE>>)
***UNREC. <RHS> = "M::=''CO(SIZE:<[+INT] <W!BYTE>>)"
←M:=''CO(SIZE:<[+INT] <W//W\W//\!BYTE>>)
FE
←P//PROCESSOR:=''CO(TP//AVG←EXEC←TIME:<[+REAL]>)
FE
←* INSPECT THESE;;  @CO;; @M;; @P
''FE(COST:<[+INTEGER]>;REL:<Ø.Ø--1.Ø>)(M;P)
''CO(SIZE:<[+INT] <W!BYTE>>)()
''CO(TP:<[+REAL]>)()
←S//SWITCH:=''CO()
FE
←* NOW SOME SPECIFIC COMPONENTS:
←M.P:=''M(FCN:PRIMARY//PRI;Ø.8Ø;2ØØØØ;16K W;16 B/W)
REL : Ø.8Ø
COST : 2ØØØØ
SIZE : 16K W
PLEASE TYPE ATTRIBUTE FOR: 16 B/W
WIDTH
WIDTH : 16 B/W
FE
←* NOTE INFERRAL OF ATTRIBUTES FROM INDEFINITE EXPRESSIONS.
←* NOW WILL INFER FROM KNOWLEDGE JUST GAINED:
←M.SEC:=''M(32 B/W)
WIDTH : 32 B/W
FE
←* PROVE ALL THESE PARAMETERS ARE THERE;;  @M.P;;  @M.SEC
''M(COST:2ØØØØ;FCN:PRIMARY;REL:Ø.8Ø;SIZE:16K W;WIDTH:16 B/W)()
''M(WIDTH:32 B/W)()

←P.C:=''P(1.Ø\Ø\2;Ø.97;(S.MP))
TP ; 1.2
TP : Ø.97
COCOMPONENTS : (S.MP)
FE
←* THE "COCOMPONENTS" WAS INFERRED BY SPECIAL-CASE.
```

```
←S.MP:=''S(0.9;(M.P P.C K.LPT K.TTY))
REL : 0.9
COCOMPONENTS : (M.P P.C K.LPT K.TTY)
FE
←« NEED TWO MORE COMPONENT TYPES:
←K:=''CO();;      T:=''CO()
FE
FE
←K.LPT:=''K((S.MP T.LPT);NAME:'PRINTER←CONTROLLER)
COCOMPONENTS : (S.MP T.LPT)
FE
←K.TTY:=''K((S.MP T.TTY);NAME:'TELETYPE←INTERFACE);;-
T.LPT:=''T((K.LPT);NAME:'LINE←PRINTER);;-
T.TTY:=''T((K.TTY);NAME:'TELETYPE)
COCOMPONENTS : (S.MP T.TTY)
FE
COCOMPONENTS : (K.LPT)
FE
COCOMPONENTS : (K.TTY)
FE
←*SAVE OUR WR↑R↑ORK SO FAR:
←L:=FAMILY('FE);; @L
LIST
(T.TTY;T.LPT;T;S.MP;S;P.C;P;M.SEC;M.P;M;K.TTY;K.LPT;K;COMPONENT;FE)
←* SAVE EVERYTHING IN THIS LIST ON THE DISK FILES
←SAVE(L,'DEMO.PMS)
(YES ! <ANY>) DO YOU WANT FILE CLOSED? :YES

←* DID IT WORK? CHECK FILE TO MAKE SURE:
←INSPECT('DEMO.PMS,1,3)
T.TTY:FE:''T(COCOMPONENTS:(K.TTY);NAME:'TELETYPE)()
T.LPT:FE:''T(COCOMPONENTS:(K.LPT);NAME:'LINE←PRINTER)()
T:FE:''CO()(T.LPT;T.TTY)
>>>4 (<NO>!<START>,<NO>!<NULL>) :

←* CHECK OUT THE FAMILY-TREE OF CO'S:
←FAMGRAPH('FE,50)
>>>LOADING FAMGRAPH:FTG.LIB/
FAMGRAPH,FTGLG,INSERT,SKIP,FIND.MAX.DEPTH,F.M.D1,CLEANUP,
--FE----------COMPONENT-*-K---------*-K.LPT
                        !           !
                        !           --K.TTY
                        !
                        !
                        *-M---------*-M.P
                        !           !
                        !           --M.SEC
                        !
                        !
                        *-P-----------P.C
                        !
                        !
                        *-S-----------S.MP
                        !
                        !
                        --T---------*-T.LPT
                                    !
                                    --T.TTY
```

```
←* DEFINE OUR STRUCTURE TO HAVE 4 M.P'S, 2 P.C'S
←GNO OF M.P:="0--3";;  GNO OF P.C:="1--2"
0--3
1--2
←* DID WE ADD TO THE PARAMETER-SETS?;; @M.P;; @P.C
''M(COST:20000;FCH:PRIMARY;GNO:0--3;REL:0.80;SIZE:16K W;WIDTH:16 B/W)()
''P(COCOMPONENTS:(S.MP);GNO:1--2;TP:0.97)()
←* OK, NOW COMPUTE RELIABILITY OF OUR SYSTEM, ASSUMING WE NEED THESE:
←* TWO M.P'S:
←MA:=''M((SA))
COCOMPONENTS : (SA)
FE
←* ONE P.C: ;; PA:=''P(COCOMPONENTS//COCO:(SA))
FE
←* FORGOT TO SAY TWO OF M.P  ;;  GNO OF MA:="1--2"
1--2
←* ONE TTY: ;; TA:=''T(COCO:(KA);NAME:T.TTY)
FE
←* THE TTY INTERFACE: ;; KA:=''K(COCO:(TA SA);NAME:K.TTY)
FE
←* AND OF COURSE THE SWITCH: ;; SA:=''S(COCO:(MA PA KA))
FE
←* DEFINE OUR STRUCTURE AS A LIST OF ACTUAL COMPONENTS:
←SL:='(P.C S.MP M.P T.LPT T.TTY K.TTY K.LPT);; COUNT(L\L\SL)
LIST
7
←* LIKEWISE THE PATTERN-STRUCTURE:
←PL:='(PA MA SA KA TA);; COUNT(PL)
LIST
5
←* FIND RELIABILITY ;; RELIB(PL,SL,1)
>>>LOADING RELIB:RELIB.LIB/
RELIB,PMATCH,MATCH,BIND,UNBIND,FANCNT,LOCREL,BINOM,FACT,A2S,PARCOVP,SIDE
NT,
APL PA;MA;SA;KA;TA;
AFCP 1;2;1;1;1;
ASL P.C;S.MP;M.P;T.LPT;T.TTY;K.TTY;K.LPT;
AFC 2;1;4;1;1;1;1;
AREL
RELL - BEGINNING MATCHES
0.0000000
←* SOMETHING'S MISSING SOMEWHERE.  FIRST SAVE WORK:
←SAVE(FAMILY('CO,'(PL SL)),'RELIB.T1)
(YES ! <ANY>) DO YOU WANT FILE CLOSED? :YES

←* SEE IF TROUBLE IS COCO'S ;; @COCO OF MA;; @COCO OF PA;; @COCO OF SA;;
@COCO OF KA;; @ \ \COCO OF TA
(SA)
(SA)
(MA;PA;KA)
(TA;SA)
(KA)
←@COCO OF M.P;; @COCO OF P.C;; @COCO OF S.MP;; @COCO OF K.TTY;;-
@COCO OF T.TTY

(S.MP)
(M.P;P.C;K.LPT;K.TTY)
(S.MP;T.TTY)
(K.TTY)
←* FORGOT TO CONNECT M.P TO S.MP! ;; COCO OF M.P:='(S.MP)
LIST
```

```
←* 3 M.P'S, 2 P.C'S;; GNO OF MA:="1--3";; GNO OF PA:="1--2";;-
←* NOW FOR RELIABILITY ;; RELIB(PL,SL,1)
>>PLEASE TYPE RELIABILITY FOR P.C
0.98
>>PLEASE TYPE RELIABILITY FOR T.LPT
0.75
>>PLEASE TYPE RELIABILITY FOR T.TTY
0.50
>>PLEASE TYPE RELIABILITY FOR K.TTY
0.98
>>PLEASE TYPE RELIABILITY FOR K.LPT
0.93
APL PA;MA;SA;KA;TA;
AFCP 2;3;1;1;1;
ASL P.C;S.MP;M.P;T.LPT;T.TTY;K.TTY;K.LPT;
AFC 2;1;4;1;1;1;1;
AREL 0.98;0.9;0.80;0.75;0.50;0.98;0.93;
RELL - BEGINNING MATCHES
MINIMAL U.V. 2;1;3;;1;1;;
UPVECTOR 2;1;3;;1;1;; HAS RELIBE 0.0030359  TOTAL = 0.0030359
UPVECTOR 2;1;4;;1;1;; HAS RELIBE 0.0030359  TOTAL = 0.0060718
UPVECTOR 2;1;3;1;1;1;; HAS RELIBE 0.0091077  TOTAL = 0.0151795
UPVECTOR 2;1;4;1;1;1;; HAS RELIBE 0.0091077  TOTAL = 0.0242873
UPVECTOR 2;1;3;;1;1;1; HAS RELIBE 0.0403342  TOTAL = 0.0646215
UPVECTOR 2;1;4;;1;1;1; HAS RELIBE 0.0403342  TOTAL = 0.1049557
UPVECTOR 2;1;3;1;1;1;1; HAS RELIBE 0.1210027  TOTAL = 0.2259584
UPVECTOR 2;1;4;1;1;1;1; HAS RELIBE 0.1210027  TOTAL = 0.3469610
0.3469610
←* SHOW GRAPHIC ANALYSIS
←TC OF M.P:=1.7
1.7000000
←* DEFINE SOME M.P PARAMETERS: ;;TA//ACCESS←TIME OF M.P:=0.9;;-
TC//CYCLE←TIME:=1.7
0.9000000
←STRINT('(P.C M.P))
WELCOME TO STRINT.
TYPE <CR> TO SUPPRESS FN TRACE

TYPE OUTER-LOOP (ROW) EXPR:
M:=2 STEP 1 UNTIL 6
TYPE INNER-LOOP (CURVE-FAMILY) EXPR:
N:='(2 4)
TYPE EXPR'S TO PLOT, IN SNOBOL4 FORMAT (NOT PMSL!!)
1: P = P
2: C = C + 70000
3: Q = P / (C + 70000)
4:
(PLEASE BE PATIENT WHILE I COMPUTE 30 VALUES!)
PLEASE TYPE COST OF P.C
6000
MANIPULATE PLOT RANGES IF DESIRED:
-I(1 2 3)
1 0.8390465  1.7262319
2 122000  214000
3 0.0000049  0.0000089
-C(1,0.8,2.0)
-C(2,0)
-I(1 2)
1 0.8  2.0
2 0  214000
-
DSK (LPT) OR TTY? :TTY
```

```
OUTER = M  FROM 2 TO 6 BY 1
INNER = N = (2;4)
TP:0.97;TC:1.7000000;TW:0.8000000;CP:6000;CM:20000;PCN:P.C;MPN:M.P
P=P  MIN=0.8  MAX=2.0  SCALE FACTOR=0.0200000 / COL.
C=C+70000  MIN=0  MAX=214000  SCALE FACTOR=3566.6666229 / COL.
Q=P/(C+70000)  MIN=0.0000049  MAX=0.0000089  SCALE FACTOR=0.0000001 / CO
```




```
(Y ! <ANY>) WANT TO RE-PLOT SAME DATA?
NO
Ø
←SAVE(FAMILY('CO,'(PL SL),'BIGDEM.PMS)
***UNREC. <RHS> = "SAVE(FAMILY('CO,'(PL SL),'BIGDEM.PMS)"
←SAVE(FAMILY('CO,'(PL SL)),'BIGDEM.PMS)
(YES ! <ANY>) DO YOU WANT FILE CLOSED? :YES

←↑Z
(YES ! <ANY>) REALLY WANT TO QUIT? :
YES
*↑C
```