CHAPTER 2

SIMULATION SYSTEM USE

This chapter describes the characteristics of the simulation system as they appear to its users, and demonstrates the system functions that are available to them.

MASH System Characteristics

The MASH simulation system consists of an integrated set of PDP-10 subprograms that contain functions for creating initial simulation states, defining simulation runs, executing simulations, and producing outputs. Several related programs perform ancillary functions that support the above procedures. The MASH system and these related programs are connected by common access to a set of on-line data and documentation files.

Four MASH system characteristics are very important in understanding the environment in which the system operates and the structure underlying the functions provided to users. They are: (1) the use of *machine readable codebooks* for micro data definition and retrieval; (2) an on-line attribute library for cataloguing definitions of attributes that are to be created by a simulation; (3) a permanent, on-line dictionary within MASH for storage of a wide variety of entity definitions; and (4) the characteristics of the MASH user command language and the command forms available. These characteristics are described below.

Machine Readable Codebooks. The MASH system uses a system of machine readable codebooks to provide comprehensive data documentation for users of these data and a reasonable degree of *data independence* for programming tasks associated with them. Historically, data files used for social science research have been poorly documented, both for purposes of content and use. The insularity of many social science computing activities, coupled with a tradition of individual research, a reluctance to make available data not thoroughly exploited by their collector, and the lack of professional rewards for data documentation have combined to assign low priority to this activity. With some exceptions, most existing data files used by social scientists are most thoroughly -- but nevertheless inadequately -- documented within programs that have either created them or read them. Although there is evidence of use of machine readable codebooks within social science computing, e.g. [B4], [I3], [S2], their use is not widespread. The lack of promising standardization activity in this area has had the effect of producing multiple, local, incompatible, specialized, and partial standards. Further discussion of these issues appears in Appendix 3.

MASH requires that each micro data file to be read be defined by another file, its *codebook file.* This codebook is used to control the extraction of observations and values for inclusion in an initial population for simulation as well as to retrieve documentation about the data source.

Codebooks acceptable to MASH are machine readable files that may be thought of as decks of punch cards. Each card, or line, contains information about the physical or logical structure of the data file or information about a data file attribute. The codebook may be printed in a human readable form by a command from the user. Figure 2-1 contains a simplified illustration of formatted codebook output. A detailed definition of codebook structure and content and a more comprehensive example are contained in Appendix 3.

The MASH codebook is a machine readable character string file consisting of a sequence of fixed length *lines* that are recorded on punch cards, disk, tape, or any other machine sensible medium. The codebook file contains the following information about the data file it describes: (1) the record or *segment* types occurring in the file and their names, lengths, and the *flag* field and value for identifying the occurrence of each; (2) the character string format and packed binary format (if it exists) of each segment type; (3) a unique name, label, mode, and positional information for each field, or *attribute,* in each segment type; (4) the discrete and continuous ranges of *values* that each attribute can assume, with label information; (5) special data conditions; (6) free form text describing any aspect of the data file and the survey that generated it; (7) end of data indicators; and other information.

For the user, the most important property of machine readable codebooks for data definition is that they allow a mechanism for addressing objects *by name.* Each *segment,* or record, type in the data file has a segment name. For example Figure 2-1 contains a subset of the PERSON segment definition for the 1967 Survey of Economic Opportunity[1] file. Each segment of the codebook corresponds to an entity level in the enumeration unit structure. The segment names corresponding to the three levels of MASH micro population are INTUNIT (interview unit), FAMILY, and PERSON.

Each *attribute* in each segment has an *attribute name,* e.g. MILITARY. This feature is used by MASH (and can be used by any program that uses the codebook to address the data indirectly) to reference documentation, values, and positional information for attributes for a variety of purposes. Furthermore, the codebook also contains for each attribute its set of its legal values, and a short description associated with those values. For example, the value "5" for attribute WORKING corresponds to the meaning "40-47

[1] This file and its uses in microanalytic simulation activity are described in Appendix 1.

weeks worked (last year)." MASH uses this information to report not only the value of specific attributes to the user, but also their meaning.

Once data have been defined in machine readable form, it is possible to retain that documentation and attach it to its data wherever it appears. MASH retains the codebook description of all attributes that it either reads from survey or census files or creates during a simulation. Machine readable codebooks are then generated by MASH for all micro data files written for output. Thus, machine readable documentation, once introduced in the system, is used for addressing data, labelling values, and is also perpetuated through the system to document outputs.

SEGMENT 6: PERSON

NO.	ATTRIBUTE	LABEL	MODE/OPERATOR		VALUE
14.	MILITARY	NOW IN ARMED FORCES	N		
		YES	EQ		1
		NO	EQ		2
		MISS	SC	MISS	9
		NAP (FEMALE)	SC	NAP	0
		NOTES: SOURCE FOR WEEKS IS 1967 SEO FILE, SEGMENT 4, INDEX 25.			
15.	WORKING	WEEKS WORKED (CIVILIAN)	N		
		NONE	EQ		1
		01-13 WEEKS	EQ		2
		14-26 WEEKS	EQ		3
		27-39 WEEKS	EQ		4
		40-47 WEEKS	EQ		5
		48-49 WEEKS	EQ		6
		50-52 WEEKS	EQ		7
		NAP (IN ARMED FORCES)	SC	NAP	0

Figure 2.1 A Simplified Example of Formatted Codebook Output.

The codebook standard defined for and used by MASH is the result of an evolution of the concept by the author and others over the last several years. Our original understanding of the importance of the concept came during the construction of the BEAST language [B2]. The creation of the 1966 and 1967 Survey of Opportunity Research Files [Rl] provided an opportunity to test the adequacy of such documentation and its effectiveness in processing codebook defined files. Codebooks for these files were constructed [S2] and were used with substantial success for data documentation and moderate success for indirect addressing of data by operating programs.

The Attribute Library. While many attributes in an initial simulation population are extracted from a survey or census data file, others are created by simulation. These attributes may be required in the initial simulation state and must therefore be assigned prior to the start of simulation, or they may be state attributes defined during the course of simulation.

An on-line attribute library in the MASH system provides the mechanism for storing and retrieving definitions of these attributes. Just as initial population attributes extracted from a survey file have their machine readable definitions extracted from the corresponding codebook file, so the attributes to be created in the simulation process have their corresponding definition extracted from the attribute library and have their values initialized to a default value. Attribute definitions in the library are referenced by *name* just as attributes in the data file are referenced by the names that are defined in the corresponding codebook file.

An auxiliary program, ALMAN, (*A*ttribute *L*ibrary *Man*ager), provides functions for processing the attribute library. Attribute definitions may be created, modified, renamed, and deleted. Copies of existing definitions may be made and then altered. The library's catalog may be inspected selectively in a number of ways. ALMAN uses an interrogative form of interaction to obtain directions from its users, and it provides guidance regarding alternative responses to questions.

Attribute library formats for specifying descriptions are identical with the specifications in the MASH machine readable codebook standard. Thus, it is easy to extract attribute definitions in order to assemble codebooks, and it is possible to file the contents of a machine readable codebook in the attribute library.

The attribute library extends the concept of *addressing by name* to attributes not previously addressable in this manner, and completes the definition of machine readable data by codebook information.

The MASH On-Line User Dictionary. The MASH system includes a permanent dictionary facility which may be used to retain a wide variety of entities from session to session. The dictionary is essentially a system for storage of data within MASH which can be accessed both by users and by MASH during simulation runs. Users may create entities that are used by MASH during a simulation run; conversely, MASH may create entities during the simulation run -- such as generated aggregate time series -- that may be analyzed later.

The following entity types are now defined within the dictionary system:

- An *attribute* definition is an algebraic expression that defines a new micro population attribute for output. It is a formula, and does not define a value until it is invoked for a specific micro

population. It therefore defines "virtual data" that is realized by the system by a number of procedures.

- A *census* is a list of names of procedures that can be applied to a micro population for purposes of summarizing the characteristics of a micro population in aggregate form.

- A *code* definition consists of a set of intervals and associated new values. It defines a function of those intervals into the new values. This function is often called coding, collapsing, or recoding.

- A *command* definition is a MASH command that is entered into the MASH dictionary. These commands are available for execution by naming them in an EXECUTE command.

- An *equation* is an algebraic expression that is used in the definition of a macro submodel to define the procedure for computing the value of an endogenous variable.

- A *list* definition consists of a list of names or symbols. The name of a list can be used in many MASH commands as a convenient shorthand device for referencing a list of names.

- A *macromodel* definition consists of an ordered list of names of equations that are stored in the MASH user dictionary.

- A *procedure* definition consists of an ordered list of MASH commands which are stored in the dictionary. The execution of a procedure is equivalent to the sequential execution of the commands named in the procedure, in the order in which their names appear.

- A *sample* definition consists of an algebraic and boolean expression which is either true or false when applied to any micro population entity for which it is defined. Samples are used to restrict certain input, output, and computational procedures to only those micro population entities which are defined to be in the sample.

- A time *series* definition consists of an initial year for the time series, a final year, and a sequence of numeric values for all years in the range specified. Series entities are used to store aggregate annual economic and demographic time series.

- A *table* definition consists of an ordered list of time series names. It is used to group aggregate time series for organizational, display, and output purposes.

Each entity in the MASH user dictionary is uniquely identifiable by three descriptors: (1) the name of its owner,[2] consisting of from 1 to 10 MASH alphabetic characters and digits; (2) the type of the entity; and (3) the name of the entity, also consisting of from 1 to 10 MASH alphabetic characters and digits. A user may own as many entities of any type as he wishes, subject to on-line storage capacity constraints. Entities belonging to other owners may be referenced, copied, and then modified by their new owner. The owner of an entity may supersede any of his entries with new versions of that entity and may also delete any entities owned by him. An index to everyone's dictionary, one's own dictionary, or selected subsets of either is easy to obtain.

---

[2] Conceptually, the dictionary may be considered as either: (1) a single entity subdivided into sections belonging to different owners; or (2) a number of separate dictionaries, each of which belongs entirely to a single owner. In the latter case, the name of the owner may be thought of as the name of the dictionary.

Access controls on entities within the dictionary provide interlocks against inadvertent destruction of one's own entities (by using a protection feature) and of entities belonging to others. Access to entities owned by others is permitted at all times, both for display and for copying. These access rules allow cooperative model development and research without imposing a more comprehensive and potentially more restrictive access mechanism upon users of the dictionary.[3]

MASH Command Descriptions

MASH provides social science researchers and policy analysts with a high level, free form command language that includes features for: (1) entering and modifying definitions, displaying entries saved in the MASH dictionary, and performing other "housekeeping" chores; (2) creating initial populations for simulation, browsing through micro data on-line, and taking surveys of simulated future population states; (3) defining and executing macroeconomic models; and (4) specifying and controlling the simulation of a microanalytic model. Knowledge of both the syntax of the MASH command repertoire and the semantics of each command, i.e. the meaning and operational properties of the command, is important for effective use of the system.[4]

Structuring the user interface to MASH around a high level command language reflects the belief that social science research is assisted by computing tools that are discipline-oriented and directly usable by non-programmers. Some supporting evidence exists in the form of the popularity and widespread use of two such languages for data analysis, Data-Text [Hl] and SPSS [Nl]. Many analytical procedures used in the social sciences are highly stylized and can be represented in a machine interpretable syntax. Some of the syntactic forms are suggested by the author and others in BEAST [B2].

MASH functions are invoked by the user by issuing a sequence of commands. Each command directs the system to perform a specific action or series of actions. Sometimes a command specifies that an action is to be performed, but its execution is delayed until another command has been received. With a few exceptions, each MASH command begins with the imperative form of a verb. Almost all MASH

---

[3] Greater security could be provided by emulating the operating system and providing password access to individual users' dictionaries; however, this would require substantial resources to ensure that password storage was secure within MASH. An alternative measure chosen was to use the operating system's file security mechanism to prevent any access to the MASH dictionary by computer users who are not members of the specific research project.

[4] The commands described in this chapter represent the state of the MASH system at the beginning of 1975. The command repertoire has undergone constant revision and extension since the beginning of its development, and therefore these descriptions do not now accurately represent the state of the language. For the user, a more precise specification is necessary and may be obtained from existing working documents. For the interested general reader, however, the following pages are sufficiently precise and impart a flavor of the structure and richness of the command forms and functions that are available.

commands are complete English sentences, except that they are terminated with a semicolon instead of with a period.

All MASH commands except two may be preceded by a command *label*. If a command label appears, the command is first stored in the user's dictionary using the label as its name, and the command will then be executed.

MASH commands are entered through the user's terminal in free form, and may be entered on as many lines as are required to specify the command. The user will be prompted for each command with an asterisk * on the left hand side of the termina1 paper or screen. If the command is not terminated on the initial line, subsequent lines are prompted by an ampersand & to remind the user that the current command has not yet been terminated. The semicolon character ";" terminates each command. A partially entered command may be cancelled at any time by typing the sequence "escape" (or "altmode" or "prefix") followed by "return".[5]

All interaction between MASH and the user during a MASH session is recorded in a protocol file for the session. Lengthy output to the user's terminal may be suppressed with the knowledge that it is being recorded in the protocol file for the session and can be retrieved and printed at a later time. Protocol file output pages may be titled under user control.

MASH commands are constructed using a subset of the ANSI character set [U3]. Some of these characters are used to form names and other symbols, while others are used as separator characters. Appendix 2 contains a definition of these symbol sets as well as detailed descriptions of all current MASH commands.

In all MASH commands, any text that follows the terminating semicolon is treated as a comment. If the command line starts with a semicolon, i.e. the command is omitted, then the entire line is treated as a comment. This allows a user to annotate a MASH session while it is in progress. For example:

```
*; Dictionary entries have now been defined,
*; so now create initial population for simulation.
```

[5] MASH executes as a standard user program under the control of the TOPS-10 monitor system. All of the terminal conventions and rules of operation of the monitor system, such as logging in and out, invoking MASH and other user programs, correcting typing errors and manipulating files therefore apply. For a complete description of these conventions and rules, see [D2], [D3] and [D4].

All comments entered during the course of a MASH session are reviewed by a project staff member, who will respond to any questions asked by the user. Thus a user may register comments and questions during a session and expect to receive reasonably prompt feedback after the session is over from someone who is knowledgeable about the system.

Dictionary and Control Functions

Selecting a Dictionary. The MASH on-line user dictionary provides a mechanism for MASH users to define various kinds of entities that may be saved and used either later in a simulation exercise or during a subsequent MASH run. Any user may begin and use a dictionary by issuing a START command. For example, suppose a user named Webster wished to start a MASH dictionary. This is accomplished by the command:

```
START DICTIONARY FOR WEBSTER;
```

If a dictionary already existed for Webster, it would be selected for reference during the MASH session by either of the commands:

```
USE WEBSTER'S DICTIONARY;
USE DICTIONARY BELONGING TO WEBSTER;
```

All subsequent references to dictionary entries that are not explicitly qualified by another owner's name will automatically be assumed to be Webster's entries.

There is no requirement that the name of the dictionary owner be identical with the name of a person. For cataloguing purposes it might be convenient to define fictitious owners such as MODEL.1, MODEL.2 and MODEL.3 that contain entities corresponding to different model definitions. The USE command may be given at any time during a MASH run, and the new owner designated becomes effective immediately. In the absence of a USE command, the default owner is STAFF.

The DEFINE Command. The types of entities that may be stored in the dictionary fall into several classes. Perhaps the most common class consists of those entity types that are essentially lists of names; sometimes those names are names of other entities. A list is entered into the dictionary by listing the names, or symbols, in it. For example, to define a list of names of demographic attributes that might appear in a codebook, the following MASH command may be used:

```
DEFINE LIST DEMOGRAPH AS AGE, RACE, SEX, SCHOOLING, INSCHOOL;
```

The above command defines a list named DEMOGRAPH in the MASH dictionary. If a specific user's dictionary was previously selected, or "opened", the list belongs to that user; otherwise it is catalogued as belonging to the owner named STAFF.

Lists may be lists of names of any meaning, although in practice they will quite often be lists of attribute names. Several other dictionary entity types are syntactically identical to lists and are defined in the same way; they include procedures, tables, macromodels, and censuses. A *procedure* is a list of names of commands, all of which belong to the owner of the procedure. A *table* is a list of names of annual aggregate time series in the "data bank" which is a subset of the dictionary. A *macromodel* is a list of names of equations, all of which belong to the owner of the macromodel definition. A *census* is a list of summarization procedures such as cross tabulation that can be applied to micro populations.

Aggregate annual time series are lists of values, and are defined in almost the same manner. One more ingredient is required; the time range of the values must be specified. Examples of the syntax used for defining times series are:

```
DEFINE SERIES GNP FROM 1947 THROUGH 1951 AS 231.3, 257.6, 256.5, 284.8, 328.4;
DEFINE SERIES UNEMPL in 1973 AS 5.2;
```

No unit of measure or scaling factor is assumed in the MASH dictionary. Therefore, time series should be entered with a convenient scaling factor, e.g., U.S. GNP in billions, and macro model equations that use or update the time series should assume the scale factor used to record the series values.

Codes, or recodes, may also be defined within the MASH dictionary and may then be used to define new attributes. Suppose, for example, that it is desired to recode the attribute AGE into the classes: (1) under 5 years, (2) 5 through 13 years, (3) 14 through 20 years, (4) 21 to 65 years, and (5) 65 years and above. Such a code would be defined in MASH in the following syntax:

```
DEFINE CODE AGECODE AS (0-5/5-13/14-20/21-64/65-*);
```

where the symbol "*" is interpreted as larger than any positive number if it appears on the right side of the dash and smaller than any negative number if it appears on the left side of the dash. The codes associated with each of the above intervals are assigned sequentially from the left, beginning with 1. If it were desired to classify or code these intervals into the values 1, 2, 4, 7, and 9, then the following command would be used:

```
DEFINE CODE AGECODE AS (0-5=1/5-13=2/14-20=4/21-64=7/65-*=9);
```

Each code phrase, i.e. the specification between slashes and parentheses, may consist of a series of intervals of real numbers and real values; the recoded value may be an integer or a real number. For example, the statement:

```
DEFINE CODE ILLUSTRATE AS (1, 3, 5, 7, 9 = 1/ 2, 4, 6, 8, 10 = 2/ -30-0 = 3.5/
                                25-30, 40-50, 99=4.5);
```

defines a transformation that maps the odd digits into 1, the even digits into 2, the range from -30 through 0 into 3.5, and the ranges 25-30, 40-50, and the value 99 into 4.5. If a value does not appear in any code phrase, either implicitly or explicitly, it is coded into 0. Code definitions are used in conjunction with the function operator "CODED.BY," which is described below.

The DEFINE ATTRIBUTE and DEFINE SAMPLE commands are used to create derived algebraic and boolean results from attributes of micro population entities. For example, suppose that an additional attribute, payroll tax, is to be calculated for output for all persons in a micro population. If payroll tax is computed as 7% of wages up to a maximum of $800, then the following command would enter the definition of a new attribute which could be computed as part of an output operation:

```
DEFINE ATTRIBUTE PAYROLLTAX OF PERSON AS MINIMUM (0.07*WAGES, 800);
```

Samples differ from attributes only in that they must be either "true" or "false" when applied to any entity at their level of definition. For example, the following definition of sample POORFAMS specifies that only those families for which the value of the attributes TOTALWAGES and OTHERINCOM does not exceed $1000 plus $800 for each person in the family -- that number of persons being the value of the family attribute NOPERSONS -- are included in the sample:

```
DEFINE SAMPLE POORFAMS OF FAMILY AS 1000 +
  800*NOPERSONS <= TOTALWAGES + OTHERINCOM;
```

A number of commands may be restricted to only those entities satisfying the conditions of a sample definition.

Both attributes and samples must be assigned to a specific level of the micro population unit, i.e. they must be defined as attributes or samples of interview units, families, or persons. Their definitions can only include attributes from their level of definition or from higher (more inclusive) levels in order to avoid any ambiguity in their evaluation.[6]

Expressions that define attributes and samples may be written in free form with operators and operands intermixed and spread out over multiple lines. The algebraic operator symbols and notation represent a mixture of Fortran, Algol, and MAD usage, with some redundancy provided to satisfy individual preferences. Table A2-1 in Appendix 2 contains an enumeration of MASH operators and their precedence, or hierarchical, levels.

In addition to the standard arithmetic operations of addition, subtraction, multiplication, and division, several operators have been included that are not commonly available. The Algol construction IF... THEN ... ELSE allows computation to be conditional upon the truth or falsehood of the expression following the IF, i.e. the "IF clause". If the clause, when evaluated, is true, the THEN clause is evaluated and the resulting value is assigned to the attribute or sample. If the IF clause is false, the ELSE clause is evaluated and the resulting value is assigned to the attribute or sample. IF..THEN..ELSE constructions may themselves be nested, allowing some complexity of structure in the assignment process. For example, suppose the previous definition of sample POORFAMS were to depend upon the value of an urban/rural attribute. A revised definition might be specified as:

```
DEFINE SAMPLE POORFAMS AS
IF URBANRURAL = 1
        THEN    1000 + 800*NOPERSONS <=
                        TOTALWAGES + OTHERINCOM
        ELSE    1500 + 1200*NOPERSONS <=
                        TOTALWAGES + OTHERINCOM;
```

To the best of the author's knowledge, the CODED.BY operator has not been previously formalized in statistical programming. This operator joins a numeric value on the left with a code definition on the right; the result is a discrete mapping, or recoding, into the set of values, or codes, that form the range of the code function. Using the code AGECODE defined in a previous example, one could define new attributes:

---

[6] For a description of a possible extension of the syntax that would permit boolean functions to be evaluated over tree structures, see [K2].

```
DEFINE ATTRIBUTE CODEDAGE OF PERSON AS AGE CODED.BY AGECODE;
DEFINE ATTRIBUTE ASCODE OF PERSON AS 2*(AGE CODED.BY AGECODE) + SEX;
```

Functions are available for use within algebraic expressions to compute absolute values, averages, exponentials and logarithms, maxima and minima, uniform and normally distributed random numbers, square roots, and other results. Functions available within MASH and their meanings are listed in Table A2-2 in Appendix 2. Since function references appearing in MASH attribute definitions, sample definitions and equation definitions are interpreted during execution, new functions are easily added to the system.

In a similar manner, the DEFINE EQUATION command specifies equations that are available for inclusion in a macroeconomic model. The Algol operator ":=" denotes the replacement of the value associated with the variable on the left hand side of an equation with the result that is computed from the right hand side. The variable on the left hand side of an equation is identified as endogenous, i.e. determined within the model, by its placement, since equations must be specified in normalized form. The scalar name *T* is reserved and represents the current calendar year during simulation. Subscript notation is used to reference current and lagged values stored in time series in the data bank section of the dictionary. For example, the following command:

```
DEFINE EQUATION INVSTACCEL AS NETINV(T) :=  MULT * (GNP(T-l) - GNP(T-2));
```

defines an investment accelerator equation in which this year's net investment (NETINV) depends upon the rate of growth of GNP measured over the two preceding years.

As the above examples illustrate, constants, scalars, and time dependent variables may appear on the right hand side of an equation, while equations may define both scalars and variables.

The DEFINE command may be used to store another MASH command in the dictionary. This command may be executed at any later time by naming it in an EXECUTE statement. Sequences of names of commands may also be defined; such sequences are called procedures. The commands named in procedures may be executed in the sequence in which they are named by naming the procedure in an EXECUTE statement.

Commands that Manipulate Dictionary Entities

A variety of MASH commands are available for displaying, altering, and removing entities in the user dictionary. These commands are described below.

The DISPLAY Command. Displaying the contents of a user's dictionary, either totally or selectively, and examining its index, are useful functions. The DISPLAY command provides a variety of alternative syntaxes to support these functions. To display the index to the entire dictionary, containing entries for all users, type:

```
DISPLAY EVERYONE'S DICTIONARY;
```

To display the index for the current owner's (user's) dictionary, type:

```
DISPLAY MY DICTIONARY;
```

To display the index to some other owner's dictionary, for example Smith's dictionary, type:

```
DISPLAY SMITH'S DICTIONARY;
```

Or, to display the index to all entities of a given type, say equations, in everyone's dictionary, type:

```
DISPLAY EVERYONE'S EQUATIONS;
```

To display the index to just the current owner's equations, or to Smith 's equations, type:

```
DISPLAY MY EQUATIONS;
DISPLAY SMITH'S EQUATIONS;
```

All of the above indices will list, for each entity, the name of its owner, its type, its name, its length (in triplets of dictionary words), the date of its creation or last modification, the date on which it was last accessed for any purpose other than displaying it, and whether it is protected against erasure or being superseded by a newer version of itself. More compact forms of the DISPLAY command and reference to other groupings of entries are also available and are described in Appendix 2.

The contents of any entity may also be displayed using another form of the DISPLAY command. For example, to display the contents of the lists named FIRST, SECOND, and THIRD belonging to the current owner, type:

```
DISPLAY LISTS FIRST, SECOND, THIRD;
```

To display Tinbergen's equation named El, type:

DISPLAY TINBERGEN'S EQUATION El;

If no owner is explicitly included in the command, the requested entities belonging to the current owner are displayed.

The display output for each entity begins with the entity's information from the index. All entity types except those consisting of algebraic and boolean expressions are displayed in a form very similar to that in which they were entered. Algebraic entities are displayed in "Polish notation," which is their internal representation within the dictionary.

Other forms of the DISPLAY command give more summary information about contents of the dictionary. To display the list of names of owners who have a dictionary, and to display the name of the owner currently in use, type:

DISPLAY OWNERS;
DISPLAY OWNER;

The words USER and OWNER are synonymous. To display summary information about the number of entries a particular owner has, and other summary data -- say, for Webster -- type:

DISPLAY OWNER WEBSTER;

Any dictionary owner may protect any entity against inadvertent erasure or being superseded. For example, the following commands *protect* and *unprotect*[7] the entities named:

PROTECT LISTS BASICATTS, MICROTS;
UNPROTECT SAMPLE AGEDMEN;

---

[7] It should be noted that protection is valid *only* for inadvertent erasure, since any authorized user may obtain access to any entity in the MASH dictionary. Security of the dictionary files is obtained by using the PDP-10 file security system to not allow access to the dictionary by other than project staff members or other users authorized by the project director.

Any attempt to erase or supersede protected entities will fail, and the user will receive an appropriate message.

Two special dictionary owners, HISTORIC and STAFF, have their entries automatically protected when they are initially created. The owner HISTORIC "owns" all historic annual time series and other related information; the owner STAFF is provided as a library area in which semi-permanent entities may be stored and referenced by several users.

An owner may remove entities from his dictionary by *erasing* them. For example, the command:

ERASE SAMPLES AGEDMEN, EASTERNERS;

erases the sample definitions named AGEDMEN and EASTERNERS belonging to the current user. If the entity is protected, it must first be unprotected.

Any user may make a *copy* of any entity in the dictionary, optionally giving it a new name. To make a copy of a list named YSOURCES and name it YSOURCES2 (possibly prior to modifying the copy), type:

COPY LIST YSOURCES NAMING IT YSOURCES2;

If the list belongs to Smith, and there is no reason to rename it, type:

COPY SMITH'S LIST YSOURCES;

to create a list named YSOURCES in the current owner's dictionary.

An owner may *rename* any of his existing entities using the RENAME command. To rename a sample named POORFAMS to FAMSINPOV, type:

RENAME SAMPLE POORFAMS TO FAMSINPOV;

Several additional commands are available for modifying MASH list-like entities, i.e. lists, procedures, tables, macromodels, censuses, and series. For all of these types except series, the commands ADD, DELETE, and REPLACE apply. Time series are modified using the ALTER command.

For the purposes of illustration, suppose the current owner owns a list named DEMOG which has been created by the command:

DEFINE LIST DEMOG AS AGE, RACE, SEX;

To substitute, or *replace*, any element of this list with another -- for example, to replace RACE by SCHOOLING -- type:

REPLACE RACE BY SCHOOLING IN LIST DEMOG;

The modified list will be identical with one that would be created by the statement:

DEFINE LIST DEMOG AS AGE, SCHOOLING, SEX;

The owner of an entity may *delete* one or more elements, or symbols, or names, from any list like entity using the DELETE command, provided that it is not protected. For example, to delete AGE and SEX from the above list, type:

DELETE AGE, SEX FROM LIST DEMOG;

One or more elements may be deleted from the same entity using one DELETE command. If all elements are deleted from an entity, the entity is erased. The names to be deleted may occur in any order, regardless of their relative position in the entity. If any of the names to be deleted do not exist in the entity, the command is aborted without any deletions occurring.

The owner of an entity may add one or more elements anywhere within a list-like entity or at its beginning or end. For example, to add INSCHOOL to the end of the above list, type:

ADD INSCHOOL TO LIST DEMOG;

Alternatively, to add INSCHOOL at the beginning of the list and to add FAMRELAT after AGE, type:

ADD INSCHOOL TO LIST DEMOG AT BEGINNING;
ADD FAMRELAT TO LIST DEMOG AFTER AGE;

Only one command, the ALTER command, is available for modifying time series entities. The command may be used to either alter current values of a time series in the dictionary or extend a series by adding more values to it. Some examples of the alter command are:

```
ALTER SERIES GNP IN 1972 TO 1047.3;
ALTER SERIES UNEMPL FROM 1965 THROUGH 1968 TO 5.4, 5.2, 4.9, 4.7;
```

The number of values entered must be consistent with the length of the time period specified. The word THROUGH means "starting with year 1 and proceeding *through* year 2", inclusive.

This completes the description of the basic commands that are used to define and manipulate MASH dictionary entities.[8]

One control function is important to the user for efficient use of MASH, the TURN statement. It is currently used for two purposes, output suppression and option selection. The first function allows a user to suppress output from MASH to his terminal either temporarily or permanently by issuing the following commands:

```
TURN ON CONSOLE;
TURN OFF CONSOLE;
```

The entire console dialogue for each run is reproduced in the session's *protocol file*. Output that is transmitted to the user's terminal will be recorded in the protocol file regardless of whether it is typed or suppressed at the console. The user may therefore select those parts of the dialogue to be suppressed with the knowledge that they are not lost.

Another use of the TURN command involves the use of options for providing various functions that are not (yet) incorporated in the command language. These options may be turned on or off either by program control or at the command language level, and may be used to include policy alternatives in microanalytic object model program modules.

Micro Population Functions

---

[8] Several additional commands are available for a "MASH librarian." They include the ORIGINATE command, which creates a new dictionary on a PDP-10 system where one did not previously exist, and the CONDENSE command which eliminates previously erased entities from the dictionary and index and reclaims their space (garbage collection, in list processing terminology).

MASH commands that apply to microdata processing may be divided into four groups: (1) those used to create initial populations for simulation; (2) those that support on-line browsing functions, i.e. searching through, examining and modifying microdata values; (3) those that survey micro populations and generate output files; and (4) those that define and control micro simulation runs.

The first three of the above groups of commands are described below. Simulation control commands are common to both micro and macro models, and are discussed later in this chapter.

Creating an Initial Population for Simulation

The creation of an initial population for simulation is an essential first step in a simulation experiment. The creation of this initial population using MASH consists of the following steps:

1. Obtaining a suitable source of microdata described by an accompanying codebook.
2. Determining which of the attributes in the survey data file are to be included in the initial population.
3. Adding attribute definitions from the attribute library for all additional attributes that are to be imputed initially or defined during the course of the simulation.
4. Specifying which micro time series are to be generated for each of the simulation entities during the simulation.
5. Determining whether subselection of enumeration units of the survey data file should occur, and if so, according to what rule.
6. Proceeding with the mechanics of creating the population in MASH internal form.

Initiating the Creation of a Micro Population. The first command that is required to begin creating a micro population is the SET BASE YEAR command. The base year, or initial year, is the calendar year that corresponds to the state of the micro population in its initial state. Thus, the command:

```
SET BASE YEAR TO 1969;
```

associates the calendar year 1969 with the population that will be created. After the first year of simulation, the updated population state will be associated with the year 1970.

The SET BASE YEAR command is the first command in the sequence of commands used to create a micro population, and the CREATE command is the last command. In between these commands, the user

may execute an EXTRACT command and one or more INCLUDE commands. Other commands such as GENERATE, IDENTIFY, and DISCARD may be executed if desired.

Specifying the Micro Data Source. The EXTRACT command is used to specify the names and locations of the PDP-10 files containing the "survey file" and its associated machine readable codebook file that will be used to create the initial population. For example, suppose that a 10 percent sample of the 1967 Survey of Economic Opportunity File, reformatted appropriately, were contained in the file named SEO10.DAT, and its associated machine readable codebook file were named CBK10.DAT. Suppose further that these files were stored on the on-line disk. To specify them as the initial population source, type:

EXTRACT FROM SURVEY FILE SEO10 DESCRIBED BY CODEBOOK FILE CBK10;

Including Survey. File and Library Attributes. The INCLUDE command specifies a group of either "survey" or "library" attributes to be included in the initial population. For example to include the survey attributes AGE, RACE, and SEX and the library attribute SOCECLEVEL, type the following two commands:

INCLUDE SURVEY ATTRIBUTES AGE, RACE, SEX;
INCLUDE LIBRARY ATTRIBUTE SOCECLEVEL;

These statements will cause the documentation for AGE, RACE, and SEX to be extracted from the survey file codebook and the documentation for SOCECLEVEL to be extracted from the attribute library. These definitions will be inserted in the machine readable codebook that will describe the initial population and that will be associated with the population throughout the microsimulation. In addition, the values of the attributes AGE, RACE, and SEX will be extracted for every person -- more generally, for every entity for which they are defined -- which is included in the initial population.

There is no restriction on the order of attribute names within an INCLUDE command or the list it references, or between INCLUDE commands. However, each attribute name can appear only once since all attribute names within the survey file codebook over all levels of definition must be unique. If there are duplications between the survey file codebook and the attribute library, only one of the duplicates can be included.

Attributes to be included within an initial population for simulation are extracted from only these two sources.[9] For survey attributes, both initial values and definitions are extracted. For library attributes, space is reserved for the values which are initially set to zero, and the definition is extracted. It is the responsibility of the operating characteristic modules -- either during initialization or during simulation -- to calculate values of attributes whose definitions were extracted from the attribute library.

The INCLUDE command can be used to reference a list of attribute names to be included rather than having to contain the attribute names directly. For example, the attributes AGE, RACE, and SEX could be included by entering the commands:

```
DEFlNE LIST BASIC AS AGE, RACE, SEX;
INCLUDE SURVEY ATTRIBUTES IN LIST BASIC;
```

Generating Time Series at the Micro Entity Level. The GENERATE command allows a user to retain one or more time series of attribute values for each entity in the initial population.

Simulating a dynamic microanalytic model produces a time series of values for each attribute of each entity in the micro population. For some purposes, the time series of certain attributes at the micro level must be obtained to evaluate the experiment. For example, the simulation of a policy that has an effect upon the time pattern of earnings of persons can best be evaluated by analyzing the patterns of earnings actually simulated over time for persons in the micro population.

The GENERATE command provides one method of obtaining such data. For example, to retain the value of the attribute WAGES for the last 5 years of simulation for every person, type:

```
GENERATE LAST 5 YEAR SERIES FOR WAGES;
```

If, instead, the highest 5 values for WAGES were required -- perhaps to compute a basis for the level of pensions at retirement -- they could be retained by typing:

```
GENERATE HIGHEST 5 YEAR SERIES FOR WAGES;
```

---

[9] Computed attributes obtained from attribute definitions stored in the MASH dictionary are only computed at the time micro output is generated; these definitions are not effective when a population is first being created.

Each of the above statements (but not both together) generates 5 additional attributes named WAGES00001, ... , WAGES00005. During the course of the simulation the values associated with these attributes will be retained as specified in the GENERATE statement.[10]

Micro series of first and lowest values may also be generated. A GENERATE command may reference a list of attribute names directly for convenience. Micro time series may be generated for both survey and library attributes. Multiple GENERATE commands may be used, and the order of occurrence of attributes within them is immaterial. These attributes are included in the initial population and have the appropriate values generated for them automatically during the course of the simulation process.[11]

Only one type and length of micro time series may be generated for any one survey or library attribute. Before the GENERATE command is invoked, each of the attributes either mentioned explicitly or addressed implicitly must have been included in the initial population being constructed by an INCLUDE command.

Subselecting the Enumeration Units. Unless otherwise specified, each enumeration unit in the survey file used creates exactly one microanalytic unit for simulation in the initial population. Subselection of these enumeration units is possible, and is controlled by entering a sample definition of interview units into the MASH dictionary and referencing it in the CREATE command. The sample definition must: (1) be at the interview unit level, i.e. all attributes within the definition must be interview unit attributes; (2) contain no computed attribute names or other sample names; and (3) contain no CODED.BY operators. For example, if REGION and IUWEALTH were interview unit attributes, and the region code for northeast were 1, then the command:

```
DEFINE SAMPLE RICHYANKS OF INTUNIT AS
        REGION=l .AND. IUWEALTH>=1000000;
```

defines a sample of northeastern interview units having wealth of at least $1,000,000.

Initiating the Population Creation Process. The CREATE command initiates the process of creating the initial population in MASH internal form. The CREATE command requires that the user specify a

---

[10] The number of attributes in the micro time series must be specified at the time the population is created because a fixed length space allocation is made for each entity in the population. Prespecifying the length of micro time series is a cost that is paid for the ease of addressing and storage management gained from such a strategy.

[11] As an option, the initial values of all attributes for which micro time series are generated may be used to initialize the series for each entity for which the attribute is defined.

name for the population to be created. MASH micro populations are referred to by name, which is an integer between 1 and 99.[12] For example, to create a micro population and name it "57":

```
CREATE POPULATION 57;
```

A simulation unit is created from each enumeration unit in the survey file named in the preceding EXTRACT command.

Initial populations can also be created for only a subset of enumeration units in the survey file by defining a sample of interview units (the highest level of the simulation unit structure) and restricting the CREATE command to units for which the sample definition is true. For example, the following commands could be used to obtain a population from only the rural west:

```
DEFINE SAMPLE RURALWEST OF INTUNITS AS RURAL=2 .AND. REGION=4;
CREATE POPULATION 57 USING SAMPLE RURALWEST;
```

The sample definition may contain random number functions, and the command has an option for selecting many different sequences of random numbers.

The execution of the CREATE command: (1) extracts included attributes from the survey codebook and attribute library; (2) creates the initial simulation population's codebook and constructs the internal format of the initial simulation population data; (3) initializes the value of library attributes to zero; (4) extracts survey file data and creates simulation units; and (5) applies an initializing set of operating characteristics to the initial population thus formed. At each step, a message is typed informing the user of the progress of this operation, since for large populations this operation can be lengthy.

The application of an initial set of operating characteristics to the population units uses the same logic as simulation for sequencing the computation, except that special initialization characteristics are used. These characteristics are dependent upon the source of data and must be coded to match the characteristics of the survey data file. At the present time, two sets of initializing characteristics are present in the system, one set for each of the populations described in Appendix 1. The name of the data file contained in the survey file's machine readable codebook is used to select the proper set of initializing characteristics.

---

[12] Since each micro population consists of 10 PDP-10 files that can assume arbitrary names, there can be more than 99 populations existing at one time. The effective restriction that this naming convention imposes is that there can be no more than 99 potentially active populations catalogued in the same user disk area at the same time.

On-Line Browsing Through Micro Populations

The increased computational capability offered by modern digital computers has given social scientists substantially greater power for data analysis, but often at the cost of increased separation of the investigator from his data. Phrased differently, while computers can mechanically process large amounts of data in complicated ways, they generally do not allow the investigator to get a "feel" for those data. Interactive computing environments offer substantial promise in restoring some intimacy between investigator and data, and systems that allow close contact with data such as [B6], [G4], [Ml], [M5], and [R2] are beginning to be used.

With the objective of restoring the "feel" of data to an investigator, a set of commands have been included within MASH that allow on-line examination of entities in a micro population. These commands allow a user to examine the contents of a population and make modifications to it. An important aspect of on-line browsing is that it may be intermixed with on-line simulation, i.e. the browsing commands and simulation commands may be used jointly for on-line examination of the effects of object model operating characteristics. This interaction will be discussed later.

Use of the browsing commands requires some knowledge of how a micro population is structured within MASH. For each population, each micro entity at each level is assigned a *name.* These names are chosen to be the positive integers starting with 1, for ease of reference, Thus, a newly created micro population of households will contain interview units whose names are 1, 2, ... , families whose names are 1, 2, ... and persons whose names are 1, 2, ... Furthermore, the data for each entity are stored in a specific (logical) *address;* we say that an entity *lives* at a specific address. Addresses are also chosen to be positive integers for ease of reference; there are interview unit addresses 1, 2, ... , family addresses 1, 2, ... , and person addresses 1, 2, ... When a micro population is created, all of its entities are named sequentially at each entity level and assigned to specific addresses, also sequentially at each level.[13]

There is no necessary relationship between family names and addresses or between family addresses and membership in interview units. Instead, cross-reference information is generated that describes the structure of simulation units within the population. There are membership lists that contain the names of all families in each interview unit and the names of all persons in each family. Address lists contain the current address of each interview unit, family and person in the population. Containment data include for

---

[13] The specific entity level names *interview unit, family,* and *person* are used in this discussion of the MASH browsing functions to illustrate the uses to which the commands can be put. These functions are applicable to any (up to) three-level micro population, regardless of the substantive nature of the unit of simulation.

each person the name of the family containing the person, and for each family the name of the interview unit containing the family. These membership lists, address lists, and containment data define the basic structural relationships between micro population entities.

As demographic processes are applied to the population during a simulation experiment, the structure of the initial population will change. New names will be assigned to new births, and the data describing the newly born child, including "inherited characteristics," will be stored in a new person address assigned to the child. Marriages and divorces will often cause a new family and a new interview unit to be created. Deaths will remove a person from the population; sometimes the person's family and interview unit will be left empty and will also be removed.

For each structural change in the micro population, the cross reference information is adjusted to reflect the new population structure. In addition, whenever a person changes family affiliation or creates a new family, the data for the person are moved to a new address. and the person's old address is updated with forwarding information. The person's new family name and a code denoting the reason for leaving the old family are added to the person data at the old address, and the person's old family name and a code denoting the reason for joining the new family are included in the person's data at the new address.[14] After the person has "moved," the data at the person's old address are preserved indefinitely by the system. Thus, every structural change in the micro population generates a genealogical trail that can be reconstructed by both user commands and by inter-unit operating characteristics that require that connections be made between related persons who are not necessarily still members of the same interview unit. One use of this genealogical information will be to provide a mechanism for the distribution of assets (inheritance) when a family is dissolved due to the deaths of all its members.

Selecting a Population. Since a user may have created a number of micro populations, the first browsing command should identify the population of interest, so that it may be "opened", i.e. made available for inspection. To "open" a population for browsing, type:

BROWSE THROUGH POPULATION popnumber;

---

[14] These additional historical and genealogical features depend upon the following attributes being present at both the family and the person level: (1) the reason the entity joined the unit; (2) the reason the entity left the unit; (3) the year the entity joined the unit; (4) the year the entity left the unit; (5) the entity's previous address; and (6) the entity's forwarding address. Attribute definitions for these attributes are in the attribute library, and must be included in a population when it is created. Operating characteristics that affect the population's structure update them when required, and other commands reference them also, even though they are generally invisible to the user of the system.

If a microsimulation is already in progress with this micro population, then it is already available for browsing and it is not necessary to execute the above command.

While browsing, three pointers called *browsing pointers* identify the *current* interview unit, family and person for browsing purposes.[15] These pointers may be set in a number of ways, either explicitly or implicitly as the result of a browsing command. These pointers are set explicitly using the LOOK command. To examine any entity, the user "looks" at it, identifying it by name and level. For example:

```
LOOK AT INTUNIT 34;
LOOK AT FAMILY 245;
LOOK AT PERSON 78;
```

The LOOK command sets the browsing pointer to the entity type that it references. Associated entities may be found by using the EXHIBIT command described below.

Each micro entity that has been created and is still a member of the population "lives" at some address. To determine the address currently assigned to any existing micro entity, the user asks *where* the entity is. For example: [16]

```
WHERE IS INTUNIT 34;
WHERE IS FAMILY 245;
WHERE IS PERSON 78;
```

To determine the values of any of the three browsing points, the user asks *who* the current entity is. For example:

```
WHO IS THE CURRENT PERSON;
```

A second set of optional genealogical attributes may be included in a micro population; they consist of the attributes FATHER (the name of the person's biological father), MOTHER (the name of the person's biological mother), SPOUSE (the name of the person's current or most recent spouse), and CHILD01,

---

[15] There are also three *simulation pointers* that identify a current entity at each level for simulation purposes, but the two sets of pointers are independent. At the present time there is no way to address the simulation pointers while browsing, but the addition of a "LOOK AT CURRENT SIMULATION UNIT;" command would not be difficult and might aid in observing the effects of simulation operating characteristics.

[16] For grammarians, an optional "?" character may be inserted between the entity name and the terminating semicolon in all commands that are actually questions.

CHILD05, ... (the name of the person's first (oldest), second, ... biological child). If these attributes are included in the micro population at the lowest (PERSON) level, then they will be updated correctly during the simulation process. The user can then use the WHO command as in the following examples:

WHO IS THE MOTHER OF PERSON 592;
WHO IS HER FATHER;
WHO ARE HER CHILD01, CHILD02;

If there is no current entity, a value, or name, of zero will be returned.

To determine the name of the entity "living" at at certain address, the user asks *which* entity lives there. For example:

WHICH FAMILY LIVES AT ADDRESS 714?;

If an entity used to live at the address specified, the system will respond that there is no entity currently at that address. If an entity ceased to exist while occupying that address, MASH will report its name and status.

The values of attributes of specific entities are ascertained by the user by asking *what* they are. For example:

WHAT IS AGE OF PERSON 495?;
WHAT IS REGION OF INTUNIT 31;
WHAT ARE ASSETS OF FAMILY 78?;

If an entity is specified in the WHAT command, the browsing pointer at that level is set equal to the name of the entity named. Thus, after the above three commands have been executed, the current interview unit, family, and person are named 31, 78, and 495 respectively, regardless of who the current entities were previously. If an entity is not specified in the WHAT command, for example:

WHAT ARE WAGES?;

then the value of the named attribute for the *current entity* at the level for which the attribute is defined is displayed for the user. For example, the above command, if executed after the previous three, would return the value of the attribute WAGES for person 495, assuming that WAGES is a person attribute. Both the

numeric *value* and its associated *codebook label* are typed, thus making the response more intelligible in the case of coded noncardinal attributes.

If micro time series attributes have been defined for an attribute, e.g. the last 5 values of the attribute WAGES, then subscript or time series notation may be used to address the series. For example, the command:

WHAT ARE WAGES (2) OF PERSON 64;

retrieves the second element of the micro time series WAGES for person 64, which after year T of simulated time corresponds to the attribute's value in year T-l. [17]

Modifications to the current values of attributes of micro population entities may be made by specifying the *change* to be made. To specify a change, the attribute name, entity identification and new value must be specified; knowledge of the specific value presupposes having an accurate codebook from which to determine the meanings associated with possible new values. For example, to change the AGE attribute of a person in the population, type:

CHANGE AGE OF PERSON 284 TO 51;

The value entered will be compared with the list of legal values in the population's codebook. If the value is legal, it will be confirmed with the label associated with it; thus the user receives both numeric and textual confirmation of his change. If the specified value is not included in the codebook for that attribute, then the user is informed and asked whether the list of valid codes and labels for that attribute should be displayed.

Using the codebook attributes described in Figure 2-1 for illustration, the following dialogue might take place between MASH and an investigator interested in the attributes WORKING and MILITARY:

*WHAT IS WORKING OF PERSON 71;

4 ... 27-39 WEEKS

---

[17] The meaning of WAGES(2) depends upon the GENERATE command that was used to create it. The first attribute of a micro time series, e.g. WAGES(l), could be the highest, lowest, oldest or newest value defined for the corresponding attribute depending upon how the population was created.

*WHAT IS MILITARY?;
2 ... NO

*CHANGE WORKING TO 0;

WAS 4 ... 27-39 WEEKS
NOW 0 ... NAP (NOW IN ARMED FORCES)

*CHANGE MILITARY TO 1;

WAS 2 ... NO
NOW 1 ... YES

Locating Entities with Specific Characteristics. Unless an investigator has good *a priori* knowledge of his data or wishes to examine the effects of his operating characteristics on specific micro units, he will be more interested in examining population units having specific characteristics rather than the first, second, or third such unit in sequence. The FIND command may then be used to locate entities having specific characteristics. For example, to find the first person in the micro population older than 65, type:

FIND FIRST PERSON WITH AGE>65;

If, after examining this person, another such person is desired for examination, type:

FIND NEXT PERSON WITH AGE>65;

If the beginning of the population has been examined previously, and the investigator wants now to browse further along in sequence in the population -- say after the first 500 interview units -- he types:

FIND FIRST PERSON STARTING AFTER INTUNIT 500 WITH AGE>65;

Attributes used in a FIND command must be associated with the entity that is being searched for.

The search may be conducted for an interview unit, a family, or a person. Searching is performed sequentially through interview units in order of increasing interview unit name. If the search is for a family or person, the interview units which are searched as a group in the above order are each searched internally in "left list order," i.e. by family, and within family, by person. If the search is successful, then all three

browsing pointers are redefined. The pointer at the search level is set to the name of the entity that was successfully located. Browsing pointers at higher levels, if any, are set to the name of the containing entity, and browsing pointers at lower levels, if any, are set to the first unit in the contained entity. If the search fails, the browsing pointers are unaltered.

The EXHIBIT command is used to exhibit the structure of a micro population entity, its genealogy[18], or its history. Examples of the EXHIBIT command are:

EXHIBIT STRUCTURE OF FAMILY 64;
EXHIBIT GENEALOGY OF PERSON 1761;
EXHIBIT HISTORY OF PERSON 322;

The STRUCTURE option describes the structure of the interview unit that the entity is contained in, if it is not an interview unit itself. The names and addresses of all families and persons currently contained in the same interview unit as the entity named are listed. If no entity name is specified, the current entity (to which the browsing pointer at that level is pointing) is used as the entity name. EXHIBIT STRUCTURE resets the browsing pointers so that they point to the interview unit, family and person referenced in the command. If the command specifies the family or interview unit and the exhibit level, then the browsing pointers for the lower levels are set to the first entity contained in the entity one level higher.

The HISTORY option traces the history of an entity at any level of the simulation unit structure through its various moves. Each time that a significant demographic change occurs for a micro unit, the entity moves to a new address and maintains a trail of both forward and backward pointers. The HISTORY option traces those pointers and displays the time and reason for each entity move.

Population Surveys and Censuses

The results of a simulation exercise using a microanalytic model are often best captured in the form of either surveys of individual entities or tabulations representing specific aggregations of interest to the experimenter. Neither of these two forms of output are available from aggregate models.

---

[18] The GENEALOGY option is applicable only to persons and has not yet been implemented. It will describe -- to the extent that information exists in the micro population -- the "family tree" of which the person is a member.

Microanalytic simulation processes are capable of producing a wide variety of micro data output for analysis. These outputs include cross-section samples and censuses at one or more points in time during the simulation process, time series of attributes of micro population entities, portions of simulated "life histories" of individuals, and summary information for various population subgroups.

The basic microdata output commands in MASH are the CONDUCT, OBTAIN, AND PRODUCE commands. They allow a user to take a survey or a complete census of any micro population, including an initial population that has been extracted from a three level, hierarchically structured survey data file as described in the previous chapter. The output produced by execution of these commands may consist of either (1) a rectangular data file whose observations are either interview units, families, or persons in the micro population and whose variables are attributes of that entity level, its containing entity, if any, and the first entity contained in it, if any; or (2) a hierarchical data file consisting of observations having three hierarchical levels corresponding to the three levels of the unit of simulation and containing a segment for each entity at each level of the unit. The output file produced is described by the contents of a machine readable codebook file which is produced at the same time. Output files created using this mechanism may be read by a wide variety of other data analysis programs on the PDP-10 computer, and they may be exported to computer systems of other manufacturers.

For example, suppose that a small demographic survey were desired of the population whose name is 57, and that attributes AGE, RACE, SEX, and SCHOOLING were to be extracted for the entire population. The user first selects two names, one for the survey file and the other to name the codebook that will be created and will define that survey file. These names are used in the PRODUCE command. For example, the user might enter:

PRODUCE SURVEY FILE DEMOG AND CODEBOOK FILE DEMCB;

Two PDP-10 files will be produced at the conclusion of the specification of the survey: (1) a file named DEMOG.DAT containing the survey data, and (2) a file named DEMCB.DAT containing a machine readable codebook describing the structure and contents of DEMOG.DAT in accordance with the standard described in Appendix 3.

The attributes to be included in the survey file are specified by one or more OBTAIN statements. For our example, the user could enter:

OBTAIN ATTRIBUTES AGE, RACE, SEX, SCHOOLING;

Or, alternatively the attributes to be extracted for the survey can be placed in lists of attribute names, e.g.:

```
DEFINE LIST Al AS AGE, RACE, SEX, SCHOOLING;
OBTAIN THE ATTRIBUTES IN LIST Al;
```

Multiple OBTAIN commands can be used to build a complete list of all attributes which are to be extracted.

After the PRODUCE command and all OBTAIN commands have been specified, the CONDUCT command is used to initiate and carry out the survey. For our example, the command:

```
CONDUCT SURVEY OF PERSONS IN POPULATION 57;
```

conducts a survey of all living persons in the population named 57. The survey file will be rectangular and will contain one observation for each living person in the population with the current values of the person's age, race, sex and education attributes.

Surveys may be taken of subsets of a population. For example, to restrict the above survey to include only men who are more than 65 years old, it is sufficient to enter the commands:

```
DEFINE SAMPLE OLDERMEN OF PERSONS AS AGE>65  .AND.  SEX = 1;
CONDUCT SURVEY OF PERSONS IN POPULATION 57 USING SAMPLE OLDERMEN;
```

Three-level hierarchical surveys are obtained by omitting any reference to specific level in the CONDUCT command. For example, to survey REGION and URBANRURAL at the interview unit level, FAMINCOME and FAMWEALTH at the family level, and AGE and WAGES at the person level, it is sufficient to enter the commands:

```
PRODUCE SURVEY Sl AND CODEBOOK Cl;

OBTAIN ATTRIBUTES REGION, URBANRURAL,
        FAMINCOME, FAMWEALTH, AGE, WAGES;

CONDUCT SURVEY OF POPULATION 57;
```

Other options of the survey process include the ability to survey dead persons, specify the format of the survey output file more exactly, and choose different random number seeds for stochastic functions that can be included in sample definitions.

Both original population attributes and computed attributes may be collected by a sample survey. Computed attributes are those that have been entered in the user's dictionary by a DEFINE ATTRIBUTE statement. These attribute names may be included in OBTAIN statements. Their definitions will then be retrieved from the user's dictionary and will be evaluated for each entity included in the sample. Computed attributes may in turn require that other computed attributes be fetched from the dictionary if they appear in the attribute's definition. Such intermediate attributes will be evaluated whenever their values are required but they will not appear in the survey unless they are explicitly named in an OBTAIN statement. [19]

Surveys may be conducted on a population at the time it is created, and at the end of any or all years of simulation. More than one survey may be conducted at one point in simulated time. Since the entity name and the simulated year are recorded on each line of the survey files produced, panels consisting of life histories may be obtained by combining survey files conducted after each year of simulation -- for the same sample within the same population -- and sorting them by entity name, simulated year, and line number within observation.

The CONDUCT command provides a basic "escape function" for MASH, since it allows a user to export the microdata to other programs for further processing and analysis. The existence of such a function relieves MASH of the sole responsibility of providing internally many statistical procedures. SPSS [Nl], Data-Text [Hl], PSTAT [B7], and several of the BMD series [D5] are among the well known and widely available programs that may be used to process survey files produced by MASH. Nevertheless, there may be advantages in providing some analytic functions within the system, both for reasons of efficiency and to include them within the interactive nature of the system. Some of these functions are discussed later.

Selective cross-tabulation and multiple linear regression are initial candidates for implementation within the system. The intended course of implementation is to define: (1) a COMPUTE command encompassing a variety of possible analytical procedures; (2) a CENSUS entity that is a list of names of

---

[19] The values of computed attributes (those whose definitions appear in the user's dictionary) are implicitly defined for all entities at their level of definition, even though they are not stored explicitly with the other data for those levels. Such attributes are called *virtual attributes* which contain *virtual data* as described in [Fl]. When a survey is taken (in computational time) the user who takes the survey need not be aware that certain attributes are defined by stored values (real) and others are derived from a definition (virtual). Within MASH, these virtual attributes are realized, i.e. made real, by the act of taking a survey. At other times during a MASH session -- given the current level of implementation -- they are non-existent. In the long run, such virtual attributes are expected to play a more important role in the system.

COMPUTE commands that are to be executed concurrently; and (3) a TAKE CENSUS command that allows scheduling censuses. Such a combination will allow substantial flexibility in output specification while providing economies inherent in minimizing passes over the population data files. Proposed syntax rules for these commands are included in Appendix 2. The CANCEL command will be extended to permit modification of the census taking agenda. These features will substantially enhance the system's ability to provide user specified output with no intermediate effort.

Macromodel Commands

Macroeconomic models are specified in MASH by a set of equation definitions, a set of exogenous and endogenous variable names, initial values for all endogenous variables, and time series inputs for all exogenous variables. Model variables may be *associated with,* or bound to, time series in the data bank just prior to simulation. Selective *tracking,* or subseries substitution, is permitted prior to and during simulation. Parameters may be specified as literal names in equation definitions, and may be initialized by searching parameter lists in the MASH dictionary. Results of macromodel simulations may be displayed as the simulation proceeds and in table form after it has been completed.[20]

The Samuelson accelerator-multiplier interaction model [S3] is used in this and the next chapter to illustrate the use of MASH commands for specifying macroeconomic models. The model contains three equations: (1) national income *y* is the sum of governmental expenditures *g*, consumption expenditure *c*, and induced private investment *i*; (2) consumption expenditure is proportional to national income, lagged one period; and (3) investment is proportional to the first difference of consumption expenditures. The model may be expressed mathematically by:

$$y_t = g_t + c_t + i_t \qquad (1)$$

$$c_t = \alpha y_{t-1} \qquad (2)$$

$$i_t = \beta(c_t - c_{t-1}) \qquad (3)$$

Only government expenditure is exogenous to the model. The above set of equations can be reordered into the order (2), (3), (1) to form a recursive model.

---

[20] Both the author and MASH owe a substantial intellectual debt to Mr. Mark Eisner and the TROLL econometric modelling system [M1] for many helpful discussions and ideas regarding computing tools for simulating macroeconomic models. In particular, model specification and solution strategy and the tracking mechanism used by MASH benefitted substantially from Eisner's early work with TROLL.

The equations in the above model are easily defined and catalogued in the MASH dictionary by the following three DEFINE commands: [21]

```
DEFINE EQUATION NATINCOME AS Y(T) := C(T) + G(T) + I(T);

DEFINE EQUATION CONSUMP AS C(T) := ALPHA * Y(T-l);

DEFINE EQUATION INVEST AS I(T) := BETA * (C(T) - C(T-l));
```

The macromodel, which we name SAMUELSON, is defined by naming the equations within it in a recursive order:

```
DEFINE MACROMODEL SAMUELSON AS CONSUMP, INVEST, NATINC;
```

The equations named in a macromodel definition should belong to the same owner in the MASH dictionary as the macromodel does.

---

[21] Ease of definition for this model follows from the fact that it is totally recursive. Since equation definitions allow any algebraic expression in predetermined and exogenous variables to appear on the right hand side, the specification of any recursive model is straightforward. There are two extensions to these rules that extend the scope of models that can be embedded within MASH:

(1) A GOTO function is used to provide some assistance in specifying blocks of simultaneous equations and solving them iteratively. The GOTO function has one argument, the name of another macromodel equation. Execution of the GOTO function transfers control to the named equation. Thus an equation such as:

```
DEFINE EQUATION LOOP1END AS DUMMY
IF ABS (ERRORNOW - ERRORLAST) <.0001
   THEN 0 ELSE GOTO (LOOP1BEGIN);
```

may be used to control an iterative computation; ERRORLAST and ERRORNOW are the results of two iterations, and when they differ by more than .0001 in magnitude, control is transferred to LOOP1BEGIN which initiates another iteration.

(2) An endogenous variable appearing on the *left hand side* of an equation may refer to the current period being simulated, e.g. Y(T), or it may refer to either a past or a future time period, e.g. Y(T-k) or Y(T+k), k=1,2, ... Calculating values for future periods could be useful in modelling such phenomena as consumer expectations, in which a macro model would be used to predict future values of national income which would then enter into consumer expenditure decisions at the micro level. Calculating values for past periods would allow models to record revised figures for aggregates based upon data computed at a later time period.

The PREPARE command is the first command to be given to *prepare* a macromodel for simulation. It includes a macromodel name, and the number of the microsimulation pass *after which* the macromodel will be simulated.[22] For example, to prepare the model SAMUELSON for simulation, type:

PREPARE MACROMODEL SAMUELSON;

If the definition of the macromodel *and* the definitions of all of its equations belonged to another owner, say to Smith, then the model would be prepared by typing:

PREPARE SMITH'S MACROMODEL SAMUELSON;

As combined macro-micro models grow in complexity, it may be desirable to interleave several macro submodels between micro submodel passes. The PREPARE command can be used to prepare a number of macro models, one for execution after each pass of the micro submodel. For example, to prepare macromodel FIRST belonging to the current owner for execution after pass 1, and to prepare the macromodel NEXT belonging to Smith for execution after micro pass 3, type:

PREPARE MACROMODEL FIRST FOR PASS 1, SMITH'S NEXT FOR PASS 3;

In the above example, all equations contained in FIRST belong to the current user, and all equations contained in NEXT belong to Smith.

A macromodel may have any number of *scalars*, i.e. scalar variables having no time dimension, contained within it. These scalars may be defined in three ways. First, the left hand side of an equation definition may contain either a time dependent variable or a scalar variable; a scalar variable appearing on the right hand side of some equation may have been defined by appearing on the left hand side of a previous equation.

Second, the user may choose to maintain his own list of parameter definitions and initialize scalar values by *searching the parameter list.* The SEARCH PARAMETER LIST command performs this function for the user. A *parameter list* is an ordinary MASH list of symbols, with the exception that the names of symbols in the list are paired; the first element of the pair is a scalar variable name, and the

---

[22] The process of applying all microanalytic operating characteristics to the micro population may be organized in several processing cycles, each of which applies a subset of the operating characteristics to the entities in the micro population, in a known order. This organization is described more fully in chapter 3.

second element is its associated value.[23] For example, the following command defines a MASH list that can be interpreted as a parameter list:

```
DEFINE LIST BASICPAR AS PI, 3.14159, BETA, 0.075, MULT, 3.42, AUTOREGC, 0.3,
        LABORSHARE, 0.71, ALPHA, 0.92, ADJ, -2.3;
```

Suppose that, after having prepared model SAMUELSON above, the scalars in the model were to be initialized with the values in parameter list BASICPAR. The command:

```
SEARCH PARAMETER LIST BASICPAR;
```

would retrieve the list BASICPAR and match each name in it with the name of every undefined scalar in the macromodel(s) prepared previously. If a match occurred, the next element of the parameter list will be assigned as the scalar's initial value. The above command would therefore set BETA to 0.075 and ALPHA to 0.92.

As many searches of different parameter lists as desired may be performed in sequence. However, the first name match initializes the value of the scalar; subsequent matches are ignored. Parameter lists belonging to other owners may also be searched.

As mentioned previously, MASH provides numeric options -- which may be selected by the user -- for executing various functions that are not (yet) incorporated in the command language. One such option types the name and value of each scalar defined during a search procedure on the user's terminal.

Finally, if the value of a scalar is undefined at the time it is required for evaluating an expression, the user is asked to supply its value on-line with the request, for example: [24]

```
PLEASE ENTER ALPHA:
```

---

[23] Since a parameter list is syntactically no different from any other MASH list, any use of the CHANGE command to alter either names or values should make no assumptions that pairing of symbols is automatic or that values are unique.

[24] At present the only acceptable response to this inquiry is to enter a numeric value which is to be associated initially with this parameter. A desirable extension would provide for an alternate response -- a search command naming a parameter list. Oversights in the command sequence could then be corrected without resetting the sequence of computation back to the beginning of the simulation.

The ASSOCIATE and TRACK commands provide a flexible way to link variables in the macromodel with time series in the MASH time series data bank. ASSOCIATE binds variable names, for which there may be several alternative data series, to time series names, while TRACK provides for temporary substitution of input values to the model.

In the absence of any ASSOCIATE or TRACK commands, the time series associated with a macromodel variable is assumed to have the same name as the variable and to belong to the current user.[25] Thus, for macromodel SAMUELSON above, it is assumed that the user who simulates with this model has time series in his dictionary named Y, C, I, and G. For example, whenever a value for G(T) is required to evaluate equation NATINCOME, an attempt will be made to retrieve it from a time series named G in the current user's dictionary. The same logic applies to Y, C, and I, although new values will also be calculated and stored for these time series. Thus, the default link between a macromodel and the dictionary of a user is that the names of the variables and time series are identical.

The ASSOCIATE command allows a user to associate, or bind, any macromodel variable with any time series in his dictionary. For example, the command:

```
ASSOCIATE G WITH GOVSPENDl;
```

will have the effect that whenever a value of G(T) is required for calculation, the value of GOVSPENDl(T) will be retrieved from the user's data bank and used instead. Likewise, if the following command were executed:

```
ASSOCIATE Y WITH NIUSA;
```

then values of the macromodel variable Y would be retrieved from the current user's series NIUSA, and values of Y that were computed by the macromodel would be stored in NIUSA. Association therefore sets up a two-way communications path between a user's own time series and variables in a macromodel.

Associations are also possible between macromodel variables and time series belonging to other users, provided that the variables are exogenous in the model. Such associations should not be specified for endogenous variables, since the association would result in a modification of the contents of a time series

---

[25] In the absence of any association or tracking commands referencing an endogenous variable, values for that series in the data bank will be replaced by the new values for the variable as they are computed. To avoid only partially overlaying a previously computed series by values of an endogenous variable, an option is available for first erasing the series associated with endogenous variables when the series is first referenced during or before the initial year of simulation.

belonging to someone else. For example, if the desired values of the exogenous government spending variable are stored in Smith's time series named FRUGALGOVT, then the command:

ASSOCIATE G WITH SMITH'S FRUGALGOVT;

will cause the values of Smith's series FRUGALGOVT to be used as exogenous values for variable G.

More than one association may be specified in an ASSOCIATE command; associations may be strung out in the same command. For example, the last two examples could be combined in the command:

ASSOCIATE Y WITH NIUSA, G WITH SMITH'S FRUGALGOVT;

Time series belonging to any owner may be used for association with exogenous variables. If a series belonging to HISTORIC is used for this purpose, the possessive "'S" may be dropped from the command.

Multiple ASSOCIATE commands may be executed for a macro model simulation. In addition, new associations may be specified during the execution of a simulation. Associations are stored in order of entry and searched in reverse order; thus new associations supersede old associations that reference the same variable.

Tracking provides a mechanism for the substitution of an alternate series of values for endogenous variables other than those previously generated by the macro model. Suppose, for example, that after the macromodel SAMUELSON was prepared, the following command was executed:

TRACK I FROM 1942 THROUGH 1945 USING HISTORIC INVESTMENT;

In the absence of any ASSOCIATE commands, the equation INVEST would compute values of I(T) and would store them in a time series named I in the current user's dictionary. The procedure for retrieving values of I(T) for computing Y(T) in equation NATINCOME would be altered, however. If T were in the range 1942-1945 inclusive, then the value of I(T) would be retrieved from the time series named INVESTMENT belonging to the owner HISTORIC; if T were not in the range 1942-1945 inclusive, the value of I(T) would be retrieved from the time series named I belonging to the current user. The above tracking command has selectively altered the source of endogenous value variables entering into the calculation of the right hand sides of equations in the macromodel -- without altering the mechanism for storing the values computed for these variables. Tracking may also be applied to exogenous variables if it desired to piece together exogenous inputs for a variable from several sources.

Tracking may also be used to substitute selective values on the basis of other conditions. For example, the TRACK command may be used to link an endogenous variable with a time series containing values for that variable prior to the beginning of simulation. In macromodel SAMUELSON, the values of C(T-l) and Y(T-l) must be known when T is the initial year to be simulated, and the simulation will not produce these values. One way to define these initial conditions is to let these variables track their historic equivalents before the first year in which the simulation produces values. For example, suppose that HISTORIC owns actual series for national income called GNI and for consumption called CONS. Then the following commands direct the simulation system to retrieve values for Y and C from series GNI and CONS owned by HISTORIC when T is less than its value during the first year to be simulated:

```
TRACK Y BEFORE START USING HISTORIC GNI;
TRACK C BEFORE START USING HISTORIC CONS;
```

If any other owner's name were used, the possessive form would be required, e.g. SMITH'S GNI; the owner HISTORIC is the exception to this rule.

Tracking may be performed for all years during a simulation, for a range of years, or for just one year. Tracking commands may specify defaults, such as tracking from a different source before the start of a simulation; another default condition is when the value would be missing in the absence of tracking. Variables may be tracked either from a time series in the MASH dictionary or at the user's console. This latter option may be useful in a simulation exercise in which the user (policy maker) is an interactive part of the experiment.[26] Notice that one TRACK command may contain more than one tracking specification clause.

A variable may have an arbitrary number of tracking instructions associated with it. Tracking instructions are stored sequentially in the order in which they are received, and the table is searched in reverse order for interpreting them. Thus, tracking commands entered later override earlier specifications.

ASSOCIATE and TRACK commands interact in the following manner. When the value of a time series variable is required to evaluate the right hand side of an equation, the tracking table is searched in

---

[26] The ability to "track," or take input values from the user's console, has implications transcending macromodel simulation. It provides an environment for a wide variety of on-line gaming activities. For example, given the ability to share the MASH dictionary files among concurrently running MASH users, it would not be difficult to use the environment provided to construct a business gaming model in which a number of players vied for maximizing market shares, profits, or some other objective function by concurrently interacting with such a model.

reverse order of entity to find an instruction that applies to that variable in that year. If the search is successful, the content of the tracking command determines the source of the input value. If the search fails, then the associate command table is searched in reverse order of entry for the first entry containing that variable name. If that search is successful, then the corresponding time series name is used as the source of the value. If the search fails, then a series belonging to the current user and having a name identical to the variable name is used as the source of the value. If the source series is found not to exist, then the user is queried for the input value at the terminal.

When the value of a time series variable appears on the left hand side of an equation and is computed, then the tracking table procedure is bypassed, and the association table is used to determine in which time series the new value is to be stored.

The SHOW command contains options to *show* the user the current state of part or all of the current macromodel(s) that have been prepared. The TABLE option displays the macromodels prepared. The EQUATIONS option displays the names of the equations and their pointers to the storage in which the Polish string equivalent of the equation is stored. The STACK option displays the Polish string representation of the equations. The SCALARS option displays the current values of all scalar variables in the models. The CONSTANT option displays the list of constants (literal values) included in all equations. The ASSOCIATIONS option displays the association table, in the order in which associations were entered. The TRACKING option displays the tracking instructions in order of entry.

Time series in the MASH dictionary may be displayed individually using the DISPLAY command and in groups using the TYPE command. For example, suppose that it was desired to display the outputs from a simulation experiment using the SAMUELSON model described above, and that there has been no association or tracking during the run. The relevant series would then be named Y, G, C, and I. The command:

```
TYPE SERIES Y, G, C, I;
```

would produce a table of values for the four series for all years in which any were defined. Alternatively, a table could be defined as a list of these series, and the series referred to by their table name, e.g.:

```
DEFINE TABLE RESULTS AS Y, G, C, I;
TYPE TABLE RESULTS;
```

If it was desired to compare the simulation outputs with the historical series, and the corresponding names in HISTORIC's dictionary were NATINC, GOVT, CONS, and INVEST, then the appropriate command would be:

```
TYPE SERIES Y, HISTORIC NATINC, G, HISTORIC GOVT, C, HISTORIC CONS,
        I, HISTORIC INVEST;
```

Alternatively, the series list could be defined as a table and be referenced indirectly.

If no time range is included in the TYPE command, all values in all series named are displayed. The TYPE command allows typing to be restricted to a range of years specified.[27] For example:

```
TYPE FROM 1950 THROUGH 1960 TABLE RESULTS;
```

Series are generally typed in groups of five, with the year in the leftmost column. Each series is headed with its name and the name of its owner.

Although MASH does not now contain commands that perform analyses of time series data, it does provide an *export* function that allows series to be sent to other computer programs for these procedures. For example, suppose it is desired to perform analyses of the four output series from macromodel SAMUELSON. If the analyses can be done conveniently using the PLANETS program [B6], then the command:

```
EXPORT TO PLANETS IN FILE RUNl SERIES Y, C, I, G;
```

will create a PDP-10 character file named RUNl.DAT. This file will contain the series Y, C, I, and G in a form that can be read directly by the PLANETS READ statement.

The existence of this "escape function" removes much of the burden of providing time series analysis functions from MASH and allows the user greater variety and depth of software with which to do his analyses. This is especially true within a discipline oriented time sharing environment that allows rapid selection and execution of a substantial number of specialized application programs.

---

[27] The SET command contains options for specifying a default typing range and for specifying the number of series columns that are typed in parallel.

This completes the list of MASH commands for specifying macroeconomic models. The next section describes the system simulation functions and their applicability to both the macroeconomic submodel and the microeconomic submodel.

Simulation Functions

Simulation consists of the application of a set of processes, often referred to as functions or operating characteristics, to an initial simulation state. Repeated application of the simulation processes produces changes to the initial state; these changed states are referred to as simulated states. Alterations to either the initial simulation state or the set of simulation processes or both is likely to lead to different simulated states. Interest in applying simulation techniques to models is often focused upon, but not restricted to, the following: (1) exploring the characteristics of model solutions and analyzing those solutions; (2) observing how alterations in the model, i.e. the set of processes, alter the simulated states when applied to the same initial population; and (3) observing how solutions of the same model depend upon the size and composition of the entities in the initial state. In discrete simulations such as those supported by MASH, it may be of interest to observe effects both upon individual entities and upon aggregated attributes of the population.

The MASH simulation system contains a number of commands providing functions for specifying simulation experiments and controlling their execution. These commands and their meanings are specified below, with examples of how to execute a simulation experiment within MASH.

Simulation Specification. The specification of a simulation experiment consists of the specification of an initial simulation state and the set of processes included within the object model. In addition, certain outputs obtained during the simulation process may also be considered a part of the model.

Within MASH, the initial simulation state consists of an initial population representing a specific calendar year at the micro level, and a set of time series values at the macro level. At the micro level, the object model consists of a set of micro operating characteristics; at the macro levels, the model consists of a set of equations, associations, tracking commands, and scalar values. Since some of these commands may be issued during a simulation run, it should be noted that a model's specification may not necessarily be complete at the beginning of the run, nor is it unalterable during its execution.

Initial micro populations for microsimulation are created by a series of commands described in a previous section, culminating with a CREATE command. Micro populations are referred to by name; their

names consist of numbers from 1 to 99. A calendar year is associated with each micro population. For example, if the command:

SET INITIAL YEAR TO 1966;

were issued while creating a population, MASH assumes that population data represents the state of the population in 1966. The first application of simulation to the model will then produce a simulated state which is to be associated with the calendar year 1967.

Macro models, on the other hand, have no initial year included as part of their description. If a simulation experiment consisting of only a macro model is to be described, then the SET INITIAL YEAR command must be used to define the year prior to the first year for which simulated values are to be calculated. If the above command had been given prior to such a run, the value of the reserved macro model variable T would be 1967 during the first solution of the macro model.

Two basic commands are used to specify the object and duration of a microsimulation. The population to which the model's operating characteristics are to be applied must first be named, for example:

SIMULATION POPULATION IS 34;

Population 34 is now available for both simulation and browsing activity.

Then, the length of the simulation is specified by the PROCEED command. For example, to advance the micro population through simulated time through 1975, the following command is used:

PROCEED THROUGH YEAR 1975;

The user may proceed to the beginning or end of processing associated with a simulated calendar year, a specific pass through the micro population, any interview unit, family or person in the population, or any macromodel, or any equation in any macromodel. New PROCEED commands take precedence over those entered earlier in the session.

If a macro model is *prepared* before the execution of the first PROCEED command, then the simulation will include macro submodels after the micro model passes specified in the PREPARE command. If no PREPARE command has been given prior to a PROCEED, then the simulation will be a

pure micro simulation. If no micro population has been named for simulation, then the simulation apparatus would execute at most the macro models set up by the PREPARE and associated commands.

Some limited flexibility of choice of micro model exists within the command language, and additional commands are being considered. For the purpose of referencing characteristics at the command language level, each characteristic has been assigned a mnemonic name such as BIRTH, DEATH, WORKING, and TAXES. Using these mnemonics, operating characteristics may be omitted selectively from a micro model whose modules have been incorporated into MASH at "load time." After a characteristic has been dropped from a model, it may be reactivated. [28]

The ALLOW and PREVENT commands permit the user to include and exclude specific operating characteristics and other functions in an existing micro model. For example, to remove the labor force operating characteristics and tax calculations from the existing micro model, type:

PREVENT WORKING, TAXES;

To reactivate them in the model, type:

ALLOW WORKING, TAXES;

Any list of operating characteristic mnemonics may appear in either of these commands. At the beginning of a simulation run, the default condition is that all operating characteristics are allowed.

The option of removing certain characteristics from a simulation run satisfies several purposes. First, if a simulation experiment does not require that certain characteristics be executed, significant computer time may be saved by excluding it. Exercising the logical option to bypass the characteristic during execution is considerably simpler in the short run than reassembling the other operating characteristic modules and relinking and loading a new version of MASH. It also allows the same "rich" model to be used by several investigators and provides them all with the option of omitting what they do not want in their particular work. Furthermore, such partial models allow a good deal of investigation into the nature of certain simulation effects, which can be very important as the model increases in complexity. An observed result generally depends upon a number of processes in the model. By holding certain processes in control or by omitting them entirely, their partial contributions to the result may be studied.

---

[28] Future commands will address the functions of extending the modular model construction concept to the command language level and allowing the parameters of characteristics to be modified at simulation run time.

Most operating characteristic program modules are associated with summary program modules that provide two types of output, individual entity results and aggregate summaries. Individual entity results consist of one or a few lines of summary output for each time the characteristic operates upon a micro entity. Aggregate summaries generally consist of counts and tables summarizing the effects of the characteristic and are produced at the end of a micro pass, simulated year, or simulation run. Both types of output result from programming within the summary program module.

The WATCH command is used to produce micro output from one or more operating characteristics. For example, to initiate typing micro output from the birth and death operating characteristics, type:

WATCH BIRTHS, DEATHS;

Each time the birth and death characteristics are applied to a person, one or more lines of output are produced on the user's console. Each characteristic produces somewhat different output, and the numbers are in general not labelled; their interpretation depends upon specific knowledge of the program module. However, data displayed generally include the entity's name, basic demographic data whether the event did or did not occur, i.e. whether a baby was born to the woman or whether the person died, the random number upon which the decision was made, the probability of the event occurring, and other supporting numbers. Thus, a knowledge of the micro output produced by a characteristic can be quite useful in debugging it and in examining its behavior in some detail.

For each operating characteristic which results in either an event occurring or not occurring, e.g. birth and death, "success" and "failure" conditions are defined for the event. Thus a successful application of the birth characteristic to a woman is one which results in a birth; likewise a "successful" application of the death characteristic to a person is one which results in the person dying. Applications that are not "successful" are said to be "thwarted," or unsuccessful. The WATCH command can be used to display information for only those micro events that are successful or thwarted. For example, to display micro information for only those births and divorces that actually occur, type:

WATCH SUCCESSFUL BIRTHS, DIVORCES;

To display micro information for deaths that do not occur, type:

WATCH THWARTED DEATHS;

A watch command may be *cancelled* by a CANCEL command, which is explained below.

The structure of summary program modules also allows for summary information to be retained and displayed both by year and for the entire simulation. Such an option is useful, since not all information regarding the history of a process can be reconstructed from cross section data. Perhaps the simplest example comes from the death operating characteristic; it is not possible to deduce the effect of mortality upon the population last year from a cross section of this year's population, since those persons who died are no longer in the population. Each of these summary procedures must be specially programmed in a summary program module associated with an operating characteristic program module.

To display aggregate summary information retained by an operating characteristic, the SUMMARIZE command is used. For example, to obtain a summary of the marriage characteristic's performance at the end of each year, type:

SUMMARIZE MARRIAGE YEARLY;

Summaries may be obtained of any combination of operating characteristics for which summary modules exist, both yearly and at the end of the simulation run.

Both commands to display micro results and macro summaries may be *cancelled* in whole or in part using the CANCEL command. For example, to cancel watching all divorces that take place, type:

CANCEL WATCHING SUCCESSFUL DIVORCES;

To stop the yearly summary output produced by the marriage characteristic, for example, type:

CANCEL YEARLY SUMMARY OF MARRIAGE;

The user may turn on or off any summary options at any time during the course of a simulation run.

Simulation Control and Execution. Once a simulation process has been specified, its execution can be automatic if the user wishes. The PROCEED command is the basic simulation execution command, and the command:

PROCEED THROUGH 1973;

will apply the current model's characteristics through simulated time until the simulation state represents the year 1973. No other instructions are required.

If the results desired from a simulation experiment can be obtained by allowing the experiment to proceed as defined, then only the above PROCEED command is required. However, there may be times when it is desirable to view aspects of the simulation process as they occur. MASH takes advantage of its interactive computing environment to allow the user to perform *incremental simulation,* i.e. to simulate to a certain point, then perform other functions such as browsing, and then resume the simulation. Incremental simulation is supported by the conditional form of the PROCEED command.

The PROCEED command allows a user to proceed either *up to* or *through* the processing of any micro entity or macro entity in the model specified. Entities are specified by type and by name. For example:

```
PROCEED TO INTUNIT 491;
PROCEED THROUGH FAMILY 824;
PROCEED TO NEXT PERSON;
PROCEED THROUGH YEAR 1973;
PROCEED TO PASS 2;
PROCEED TO MACROMODEL 1;
PROCEED THROUGH EQUATION 7;
```

Whenever the named entity is either about to be processed or processing has concluded -- depending upon the choice of TO or THROUGH -- control is returned to the user and a prompting asterisk is typed. Simulation may be resumed by typing another PROCEED command.

In the above forms of PROCEED, macromodel numbers correspond to the number of the pass which they follow, and equation numbers are taken from the table of equations assembled from all macromodels. The SHOW MACROMODEL EQUATIONS command will display the equation numbers that can be used in the PROCEED command. The use of the optional naming device NEXT causes the simulation to proceed to or through the next entity to be processed of the type named, regardless of its name.

The simulation of micro or macro models may be interrupted at any time by entering a MASH command during execution of the simulation. [29] The interruption will be invoked either prior to processing

[29] To generate such an interrupt, it is sufficient to press the "carriage return" key, which is equivalent to entering a blank command line. The actual command text may then be entered on subsequent lines after the interrupt message has been received.

a person or an equation, depending upon the submodel being operated upon at the time the interrupt was sensed. A message is typed indicating the status of the simulation in computational time, i.e. the entities being processed and the processing pass as well as the simulated year. The simulation may be continued by executing a PROCEED command.

A simulated population, i.e. one that is the product of applying an object model to an initial population and advancing its state through simulated time, may be saved in the user's "permanent" on-line storage. In general, such a population only has meaning if the simulation is interrupted at the end of a calendar year or has proceeded to completion. The command:

SAVE POPULATION;

performs this function. The name of the population is unchanged.

Saving the population, however, is not sufficient to save the complete simulation state, since much of the information about the run is kept within the program copy being used for the run. The complete simulation state may be saved by saving both the population and the program state at that time. The command:

TERMINATE SIMULATION;

executes any end of simulation processing that may exist such as session summary calculation, and then saves the population.

A variety of other commands of the "housekeeping" class are available to the MASH user, but are of lesser general interest than those described above. For example, since the PDP-10 monitor system does not support any checkpoint facility at this time, MASH contains such a facility. For example, the command:

CHECKPOINT AFTER 1961(T) THROUGH 1964(T), 1965, 1966(T) THROUGH 1969(T), 1970;

constructs temporary checkpoints for the years 1961-1964 and 1966-1969 and permanent checkpoints for the years 1965 and 1970. Each checkpoint consists of a series of machine readable files which: (1) contain the complete status of the simulation exercise at the end of the specified simulation year, including the status of the entire micro population; and (2) can be easily restarted to continue the exercise. Temporary checkpoints

are erased by subsequent temporary checkpoints, while permanent checkpoint information remains until deleted by its owner.

Similarly, the REPORT STATUS OF command has a variety of options that allow a user to obtain information about various aspects of the current exercise. Status types include options selected, checkpoints scheduled, values of current simulation pointers, approximate position in the computational sequence, and names of macromodels prepared. A more complete description of such commands is included in Appendix 2.

The above command descriptions in this chapter illustrate the use of many of the MASH commands available to the user, but are not comprehensive or exhaustive. Comprehensive descriptions of the entire command repertoire (as of the beginning of 1975) are contained in Appendix 2. Nevertheless, it is hoped that these descriptions illustrate the scope and the comparative advantage of MASH as a user-oriented system for microanalytic simulation.