REPRESENTATION SCHEMES FOR MATHEMATICAL PROGRAMMING MODELS

Frederick H. Murphy

Edward A. Stohr

Ajay Asthana

STERN #IS-89-104

REPRESENTATION SCHEMES FOR MATHEMATICAL PROGRAMMING MODELS

Frederick H. Murphy
School of Business
Temple University
Philadelphia, PA

Edward A. Stohr
Leonard N. Stern School of Business
New York University
New York, NY 10012

Ajay Asthana
Leonard N. Stern School of Business
New York University
New York, NY 10012

Revised: June 19, 1990

Working Paper Series
STERN #IS-89-104

ABSTRACT

Because of the difficulties often experienced in formulating and understanding large scale models, much current research is directed towards developing systems to support the construction and understanding of management science models. This paper discusses seven different methods for representing mathematical programming models during the formulation phase of the modeling process. The approaches discussed are block-schematic, algebraic, three different kinds of graphical schemes, a database-oriented approach and Structured Modeling. We emphasize representations that have graphical elements suitable for incorporation in the interface to a modeling system. The different methods are compared using a common example and the transformations that allow one to go from one representation to another are discussed.

Key words: Modeling, mathematical programming, graphics

1. INTRODUCTION

Our ability to solve large mathematical programming models has improved with the introduction of new algorithms and continued advances in computer technology. The major impediment to more widespread use of these models appears to be a human one. Modeling is a time-consuming, error-prone task that is understood by only a small number of management scientists (Fabozzi and Valente [1976]). Recently, there has been an awakening of interest in the modeling process itself and in computer systems which directly support the modeler. An example of the benefits that can be obtained from the combination of a user friendly interface and powerful modeling language is given by the PLANET system at General Motors (Breitman and Lucas [1987]).

Most mathematical programming systems (Optimizers) accept model definitions in the MPS format (IBM [1975]). This consists primarily of a list of triples in the form (Row Label, Column Label, Value). Although MPS format allows sparse matrices to be represented fairly efficiently, it is difficult for humans to develop and debug models in this form, (Meeraus [1984]). Over the last few decades, a number of systems have been developed which attempt to make the modeling task easier. Early systems (Matrix Generators) were essentially procedural programming languages that helped generate files in MPS format. Some examples are GAMMA (Sander and Smith [1976]), LOGS (Brown et al [1987]), OMNI (Haverly Systems Inc. [1977]) and DATAFORM (Creegan, [1985]). Later, systems which accept problem statements in a non-procedural language were developed. PAM (Welch [1987]) and MIMI (Baker [1990]) are table-oriented languages. Non-procedural systems that take an algebraic approach include GAMS (Meeraus [1984]), AMPL (Fourer et al [1990]), GXMP (Dolk [1986]) and CAMPS (Lucas and Mitra [1985]).

Other recent modeling approaches include Structured Modeling (Geoffrion [1987]), which provides a general representation for a broad range of model types, and Netforms (Glover et al [1978], Klingman et al [1989]), which are suitable for mathematical programs that are primarily networks. Dolk [1986] has developed a system based on concepts from database management systems. Krishnan [1987] and Raghunathan [1988] have designed new modeling languages based on artificial intelligence techniques for representing domain dependent knowledge. The former uses a dialogue-driven interface controlled largely by the computer, while the latter proposes a modeling language based on And-Or graphs. Some "restricted natural language" interfaces have also been developed. Binbasioglu and Jarke [1986] develop a simple "activity-resource" language for specifying problems in the area of manufacturing production. Greenberg [1987] has developed a restricted natural language system for interpreting LP models and results. A number of systems, effective for small applications, integrate optimization with the spreadsheet paradigm (Bodily [1986]). Finally, we have built a prototype system, LPFORM, to help modelers formulate linear programming (LP) models (Murphy and Stohr [1986] and Ma [1988]). LPFORM provides a graphic interface which uses icons to represent real world objects such as inventories, machines and transportation networks. The interface is described in Ma et al [1986] and has been improved and tested by Asthana [1988].

The great diversity of existing and proposed modeling systems makes a comparative analysis worthwhile. Our objective is to review these systems from the point-of-view of the interface presented to the user. In particular, we wish to investigate various methods for representing LP problems. In the space available, we can only review some major alternatives: non-procedural programming languages, graphics-oriented interfaces and database representations. We have chosen some typical systems in each of these categories and explain them in terms of a common example. Since all viable representations must lead to unambiguous model statements, it should be possible for a model management system to transform from one representation to another. The paper outlines the steps needed to perform some of these transformations. We also introduce "compact" forms for some existing graphic representations and illustrate some interface design features that should help users cope with the complexity of real world applications.

Section 2 provides a general framework for comparing different representation schemes. Section 3 introduces an example that will be used to illustrate the different approaches. Sections 4 through 10, respectively, cover block-schematic approaches, algebraic languages, Activity-Constraint Graphs, Netforms, Structured Modeling, database representations, and the iconic approach used in LPFORM. The paper ends with some brief conclusions and suggestions for future research.

2. REPRESENTATION SCHEMES IN MODELING

In this section, we discuss the objectives of advanced modeling systems and the role of problem definition languages in helping to achieve these objectives.

A typical design for a modeling system is shown in Figure 1. Most current systems contain all of the subsystems shown in the figure in, at least, a rudimentary form. Ideally, the interface module handles a variety of input and output presentation modes. The Model Processor accepts the user's input and produces a statement of the model in a form suitable for a standard Optimizer such as LINDO (Schrage [1984]) or IBM's MPSX [1975]. A Solution Analyzer (e.g. Analyze, Greenberg [1983]) accepts the solution from the Optimizer, performs analyses, and provides reporting and online query facilities for the user. The Model Management and Database Management components provide information, access and maintenance facilities for models and data respectively (see Blanning [1982] and Date [1987] for a discussion of these components).

[FIGURE 1 ABOUT HERE]

The following are some objectives of a good modeling system:

(1) Provide a rigorous conceptual framework for problem formulation.

(2) Allow the representation of a broad range of model types.

(3) Reduce the complexity of the modeling process.

(4) Support all phases of the development and use of models.

(5) Provide model-data independence.

(6) Provide model-solver independence.

(7) Check the validity of models.

(8) Employ modern interface techniques.

(9) Integrate modeling with modern database techniques.

(10) Provide powerful computational features to help generate the data.

(11) Facilitate the reuse of previously developed models and their combination into larger models.

(12) Provide automated documentation of models.

(13) Explain model structure and interpret model results.

(14) Accumulate domain dependent knowledge over time.

The above includes the list of desirable features given by Geoffrion [1987]. The items in this list are self-explanatory except for items (5) and (6). Model-data independence, implies a separation of the statement of the structure of the model from the data that is to be used in it. Thus, the sizes of sets and values of data items can vary from run to run without changing the statement of the model. Similarly, model-solver independence implies that the statement of the model is in a format that does not depend on the requirements of any one solver or class of model.

In considering the role of representation schemes in the achievement of these goals, it is important to distinguish between external and internal representations. An external representation scheme is used by the modeler to define models and to express queries to be answered by the model. Objectives (1) through (3) simply cannot be achieved without a good external representation scheme. In addition, goals (4) through (8) are critically affected, and the remaining objectives somewhat affected, by the choice of external representation.

On the other hand, an internal representation scheme is usually invisible to the user. It is used by the system to support all of the goals on the above list. In particular, the use of knowledge representation schemes from artificial intelligence can facilitate the attainment of goals (13) and (14). The internal representation is used to generate a problem statement for the optimizer, to document the model and to act as a database for online queries concerning the structure and objectives of the model.

Although the internal and external representations are naturally related, there is every reason to believe that they should be different. The purpose of an external representation is to help the user. The formulation of models involves a mapping between real world objects and relationships and symbolic (usually mathematical) objects and relationships. This process is painful even for experts as it involves minute attention to detail. Usually, the correctness of a model can only be ascertained by trial runs involving much data processing. For nonexperts, the translation process is almost impossible because of their poor understanding of mathematical concepts such as variables and indices (Orlikowski and Dhar, [1986]). A major theme of current research is that a good external representation scheme helps users visualize the real world in conceptual terms and thereby facilitates the generation of correct models (Shneiderman [1987]). The system itself should automatically translate from the external to the internal representation scheme.

Representation schemes can be discussed in terms of four dimensions: generality, concreteness, labor-intensiveness, and interface potential. Generality refers to the applicability of the technique to a range of management science models (both within and beyond LP). The other three dimensions are aspects of what is generally referred to as "user friendliness."

The concreteness dimension measures how closely real world objects are captured. External representation schemes should be concrete in the sense that they should contain analogues of real world objects and relationships rather than mathematical objects and relationships. Internal representation schemes may be abstract since they portray the symbolic representation of the model and must, of necessity, include mathematical concepts. Evidence concerning the desirability of icons and other concrete objects that can be directly manipulated by users is quite strong (Shneiderman [1987]). Graphs can provide more concrete representations for modelers because they can reveal hidden facts and relationships and stimulate human thinking (Shepherd [1987]). A study by Carlson et al [1977], showed that decision makers seem to rely on conceptualizations and that graphs and visual scenarios helped improve decision making. The advent of low cost computer graphics technology makes interactive systems possible. For these reasons, our research has emphasized graphic representation schemes.

Labor-intensiveness (the amount of detailed book-keeping work required from the user) is a function of the complexity of the representation and is especially important for large problems. While graphics can help on the concreteness dimension, graphical representations can be too complex both to draw and to understand for the large mathematical programming models found in practice. We need to invent methods of computer support that allow users to draw high-level diagrams of major model relationships while hiding the messy details. How to provide useful forms of hierarchical abstraction that help, rather than hinder, users is a challenging area for research.

The final dimension, interface potential, measures how well the representation form lends itself to the use of advanced computer interface design features and to support for the dynamics of the user interaction. Thus, the interface should provide not only a good medium for expression of ideas, but also support problem solving strategies and other features that can help users. These include:

(1) Hierarchical definition of the problem through top-down refinement.

(2) Piece-wise model development (bottom-up development) with a submodel integration capability.

(3) Reuse of previously developed models and model fragments.

(4) Consistency and validity checking during (as well as subsequent to) the model construction phase.

(5) Memory aids.

(6) Good interface characteristics including fast response and easy revision and modification of previous work.

Of the above, we need elaborate only on item (2). By this, we mean that users should be able to define small pieces of their models in any order. The need to organize work in a strict order, to formally define objects before they are used, and to follow a rigid syntax, places an unnecessary burden on the user. As illustrated later, it seems preferable for the computer to perform the steps needed to infer missing problem components and to construct a properly ordered, consistent internal problem representation.

In the final analysis, the choice of model representation scheme will depend on both the task at hand and the particular class of user involved. Students, engineers, managers and OR experts will have different needs and individual users will probably want to use multiple representations of the same model during the course of its development and use. In this paper we hope to provide some insights into the different types of representation and how they relate to each other.

3. SAMPLE PROBLEM

To compare the different languages for representing models, a sample problem has been taken from Schrage [1987]. This is a small problem but has sufficient complexity to illustrate most of the issues involved in developing internal and external representation schemes. The problem statement is as follows:

"A farmer has 120 acres which can be used for growing wheat or corn. The yield is 55 bushels per acre per year of wheat or 95 bushels of corn. Any fraction of the 120 acres can be devoted to growing wheat or corn. Labor requirements are 4 hours per acre per year plus 0.15 hour per bushel of wheat and 0.70 hour per bushel of corn. Cost of seed, fertilizer, etc., is 20 cents per bushel of wheat produced and 12 cents per bushel of corn produced. Wheat can be sold for $1.75 per bushel, and corn for $0.95 per bushel. Wheat can be bought for $2.50 per bushel and corn for $1.50 per bushel.

In addition, the farmer may raise pigs and/or poultry. The farmer sells the pigs or poultry when they reach the age of one year. A pig sells for $40. He measures the poultry in terms of coops. (One coop brings in $40 at the time of sale). One pig requires 25 bushels of wheat or 20 bushels of corn, plus 25 hours of labor and 25 square feet of floor space. One coop of poultry requires 25 bushels of corn or 10 bushels of wheat, plus 40 hours of labor, and 15 square feet of floor space.

The farmer has 10,000 square feet of floor space. He has available 2,000 hours of his own time and another 2,000 hours from his family. He can hire labor at $1.50 per hour. However, for each hour of hired labor, 0.15 hour of the farmer's time is required for supervision. How much land should be devoted to corn and how much to wheat, and in addition, how many pigs and/poultry should be raised to maximize the farmer's profits?"

The formulation of this problem in "matrix" format is shown in Figure 2 using numeric data.

[FIGURE 2 ABOUT HERE]

We now formalize the problem somewhat by defining symbolic names for the data. Since we are concerned with the language used for the external representation, the conventions used to name the objects in the model are important. In general, long (descriptive) names, short mnemonics, and comments are all essential to good modeling practice. Short names are useful in algebraic statements. Also, unique identifiers (no longer than 8 characters) have to be supplied for the row and column labels of data coefficients in the input to many optimizers, e.g. those using MPS format. These labels can be composed by concatenating together the short names for variables, indices and data coefficients. Devising unique, meaningful short names and labels is a tedious job which lends itself well to computer assistance. Asthana [1988] suggests a suitable set of naming conventions. It is assumed that the user supplies the "long" names for all the basic objects in the model. The computer then suggests short names for the objects and data coefficients and also provides some limited help in generating descriptive comments. Figure 3 illustrates these conventions for the Farmer's Problem.

[FIGURE 3 ABOUT HERE]

4. BLOCK SCHEMATIC REPRESENTATIONS

The earliest class of LP matrix generators (such as OMNI, GAMMA and DATAFORM) contain language statements to generate the data triples for MPS statements directly. The objects to be manipulated are data tables and the end result is a long list of data triples in MPS format. Since the columns (activities) of an LP typically intersect only a few rows, practitioners using these systems generally adopt a column-oriented view of their task.

A more global "block-oriented" view of the LP matrix is useful for understanding problem structure, and seems to be the conceptual representation used by many experienced practitioners (Welch [1987]). Block-wise formulation depends on the fact that most LP matrices are composed of blocks of non-zero data structures (data transformations, diagonals and summation rows) interspersed with blocks containing only zeros. Block-schematic systems provide language statements that help the modeler place 2 dimensional arrays of data (and special 0-1 structures) into the larger 2 dimensional matrix of the LP tableau.

Figure 4 shows a block-schematic formulation of the farm problem using the language conventions of PAM (Welch [1987]). The "Matrix Schematic" table near the top of the figure provides a convenient overview of the structure of the LP matrix and is the focal point of the approach. The blocks named in the body of the matrix schematic contain homogeneous sets of data coefficients. For example, the block "FAT" contains transformation coefficients giving the number of animals produced per bushel of grain. Alternatively, a block can contain a "connection structure" of 1's or -1's that serves to connect other blocks or to sum up a number of activities; these are represented by "/1" and "/-1" icons in the figure. Note that the constraint inequalities and names of the RHS coefficient blocks of the LP matrix are included in the block schematic.

[FIGURE 4 ABOUT HERE]

The blocks are located at the intersections of column strips and row strips. The column strips are collections of similar activities that are further defined in the "Columns" table. The head of the Columns table contains the list of column strips; immediately below this is a row of short literals that are used to uniquely identify each column strip in the MPS statement. The remaining rows of the table list the domains for each column strip. The stub ({1,2,..}) serves to order the domains (indices). Similarly, the row strips are collections of similarly defined constraint rows that are further defined in the "Rows" table.

Each block of data coefficients in the matrix schematic is described in a row of the "Data" table as shown in Figure 4. The "Table" column references the data table in which the data values for each block are stored. The "Stub" and "Head" columns identify the domains that index the rows and columns, respectively, of the block in its associated data table. As illustrated in the figure, blocks that have common domains may share the same data table.

Note that the blocks that contain the data values define the row and column dimensions of the LP matrix. The connection blocks derive their size from their position in the matrix. Essentially, these blocks cause a 1 (or -1) to be inserted in the final LP matrix whenever a domain value in the column strip matches a domain value in the row strip. For example, a diagonal submatrix is generated when the domains of the column strip match those of the row strip. Other conventions in PAM allow inventory and other special matrix substructures to be generated.

The block-schematic approach is graphical, matches the way many experienced LP modelers think, supports problem solving through its essentially hierarchical approach, and should be a good vehicle for advanced interface techniques. Because it is a declarative rather than procedural language, it is also probably less labor intensive than earlier column-oriented systems. However, the representation is abstract since it focuses on the end result of the formulation process rather than on the physical system being modeled. Experts can probably "see" the real world structure in the block-schematic but this is difficult, if not impossible, for inexperienced users.

5. ALGEBRAIC REPRESENTATIONS

Using the definitions in Figure 3, the conventional algebraic representation for the Farmer's Problem is:

(1) Maximize:

$$\sum_g SP_g.S_g + \sum_a RP_a.R_a - \sum_g BC_g.B_g - HLC.HL - \sum_g HC_g.H_g$$

Subject to:

$$\sum_g HAT_g.H_g \leq AS \qquad \text{(Acres Usage)}$$

$$- HLLT.HL + \sum_a RLT_a.R_a + \sum_g HLT_g.H_g \leq LS \qquad \text{(Labor Usage)}$$

$$B_g + H_g - \sum_a F_{a,g} - S_g = 0, \quad g \text{ in Grains} \quad \text{(Grain Balance)}$$

$$\sum_g FAT_{a,g} \cdot F_{a,g} - R_a = 0, \quad a \text{ in Animals} \quad \text{(Animal Balance)}$$

$$\sum_a RFT_a \cdot R_a \leq FS \quad \text{(Floor Usage)}$$

where $HL \leq HLS$.

A number of systems have been developed which accept problem statements in algebraic form. As mentioned earlier, these include GAMS, GXMP, AMPL and CAMPS. LINDO (Schrage [1984]) allows a restricted form of algebraic input in extended coefficient form (no summations or indices). Bradley and Clemence [1987] propose an important extension to algebraic languages in which the model objects are "typed" to help model and data validation through a form of dimensional analysis.

Figure 5 gives a complete formulation of the farmer's problem in the syntax of GAMS. In GAMS, the model components such as Sets, Scalars, Parameters, Variables and Equations are specified in a fixed order using a fairly rigid syntax. In the figure, the text in small letters contains optional comments. The meaning of the problem statement should be clear to any one versed in management science. In fact, this is a major advantage of algebraic notation as an external representation scheme. Most importantly, they provide the potential for both model-data and model-algorithm independence. In the case of GAMS, these advantages are somewhat nullified because the data values and algorithm type are compiled with the model statement. It would be advantageous to support the input of data values as a separate process so that the same model can be run with different data instances.

[FIGURE 5 ABOUT HERE]

The use of algebraic modeling languages is a major step forward. They have great generality, and because they are non-procedural and concise, they are not labor intensive (at least as far as defining the structure of the model is concerned). Nevertheless, they present few opportunities for advanced interface features and involve abstract rather than concrete concepts. For these reasons, their use may be restricted to a small group of management scientists. Students with one course in LP for instance, had a hard time formulating LPs in algebraic notation (Orlikowski and Dhar [1986]).

An additional disadvantage of algebraic representation schemes is that the physical structure of the underlying problem is not made explicit. Such information can be gleaned, after the fact, from the generated matrix, and used to determine the reasons for infeasibilities (if they exist) and to explain the results of the model (Greenberg [1983]).

However, if structural information is input directly as part of the model statement, the user's comprehension of the model can be enhanced and there are additional opportunities for the system to analyze the correctness of the model during the development process (Murphy et al [1987]).

6. ACTIVITY-CONSTRAINT GRAPHS

An Activity-Constraint (A-C) graph for the Farmer's Problem is shown in Figure 6 (adapted from Schrage [1987 p.119]). Similar A-C graph representations have been proposed by Egli and Kohlas [1981] and Hurlimann [1987].

[FIGURE 6 ABOUT HERE]

Any LP can be represented as an A-C graph. There are two types of nodes. Activity nodes representing decision variables are depicted by open boxes. Constraint nodes are shown as circles. The arrows represent the effect of the activities on the resource levels associated with the constraints. If the arrow points to a constraint the associated activity provides an input to the constraint and conversely. The numerical coefficients on the arrows provide the values for the transformations. Thus, if the resource is an input (output), its level in the constraint is lowered (raised) by the value of the coefficient when the activity level is increased by one. Exogenous supply and demand values for resources are written in the circles. Constraint nodes with zero values represent flow balance equations.

An A-C graph provides an intuitively appealing representation that can help users understand, construct and check a problem formulation. The graph can be translated in a straight-forward manner into an LP matrix for input to a Solver. The coefficients on the arcs associated with each activity form the nonzero elements in its column, while the values in the constraint nodes form the RHS for the problem. However, it is usually more convenient to formulate the constraints one-at-a-time. The constraint corresponding to a constraint node is formed by adding together terms involving each activity to which it is connected. Each term is formed by multiplying the coefficient on the arc by the symbol for the variable. We follow the convention that terms on incoming (supply) arcs are positive while those on outgoing (demand) arcs are negative.

The major disadvantage of such graphs is that they are very labor-intensive, even for small problems such as that in Figure 6. (Note that the connections to the money resource in the objective function were omitted to simplify the graph). The obvious way to reduce the complexity of the graph is to replace coefficient values by array names and to use set notation to portray activity and resource types as in Figure 7A.

[FIGURE 7 ABOUT HERE]

To reduce the visual complexity, objective function coefficients are written beside their associated activities. Also, explicit upper and lower bounds on resource levels and activities are shown symbolically (rather than graphically) by including symbols for the upper and lower limits in square brackets at the relevant nodes. Finally, the index sets for the coefficients have been omitted. These can be computed as the union of the sets associated with the Activity and Constraint at either end of the arc (singleton sets are treated as null for this purpose). Note that the indices of coefficients are simply identifiers for particular values. The dimensions of the sub-matrices corresponding to the coefficients in the larger LP matrix are determined by the number of constraint and activity rows. Thus, the coefficient, FAT_{ag}, represents four non-zero values, but forms a (2 x 4) array in the tableau of Figure 2.

When there are relationships between elements with different values in the same set (as occurs with time in planning and inventory problems), it is necessary to replicate the A-C graph for a sufficient number of consecutive index values to reveal the underlying pattern. It might also be necessary to show the pattern for both the starting and ending conditions. Thus, in a finite horizon planning problem, one might depict all constraint and activity nodes for time periods 1, t-1, t and T.

Most practical LPs include a number of "side constraints" arising from policy or other requirements. Examples are generalized bounds on variables and constraints on ratios of variables such as:

(2) $\sum_g H_g \leq HLB$ and $\sum_g H_g \geq HUB$

(3) $F_{hens,corn} \geq 0.20\ F_{hens,wheat}$

$F_{pigs,corn} \geq 0.30\ F_{pigs,wheat}$

Figure 7B shows the additions to the A-C graph to accommodate these constraints. Constraints (2) are represented by the "Bushels" constraint node. A lower bound constraint can be represented as a demand node and an upper bound as a supply node. When the two are merged as in the figure, the arrow becomes bi-directional. Note that the same coefficient applies to both directions of a bi-directional arrow since constraints with the same RHS index sets must have the same LHS (Murphy et al [1987]). The "Ratios" node in the figure indicates that there is a $\geq$ 0 constraint for each member of the animals set. Using the rules given above, the coefficient R for variable F is indexed by (Grains, Animals); the values needed to capture constraints (3) are given in Figure 13 below.

While constraints such as (2) and (3) can be represented by simple extensions to the formalism, the resulting graph becomes less "concrete" since the physical flow analogy is lost. Complicated policy relationships between more than two variables are even more difficult to represent.

Many real problems have a simple enough underlying structure to be represented conveniently by A-C graphs using the above conventions and it is easy to envision the development of an advanced A-C graph modeling interface using techniques similar to those developed for CAD-CAM applications. However, building A-C graphs is likely to be labor intensive for large problems and the A-C graph representation is not general in the sense that it would have to be extended to cover non-LP problems. Finally, A-C graphs are, in general, quite abstract as they depict the end result of the formulation process rather than the underlying physical model. When the underlying problem has a simple enough form, the A-C graphs become more concrete and a closely related representation has been used very successfully as discussed in the next section.

7. NETFORM GRAPHS

Every LP model can be represented by an A-C graph because activities and constraints are logically paired by the technology coefficients. When the underlying real world problem has a network representation, there is only one arc entering and leaving each activity node. Thus, the activity nodes can be dropped without losing the uniqueness of the representation. Glover [1987] has studied such problems extensively and has developed modeling approaches for a broad variety of applications as well as a coherent set of graphical conventions (see also Glover et al [1978] and Klingman et al [1989]). Figure 8 illustrates these conventions for a network representing a modified version of the Farmer's problem in which the labor and floor constraints are disconnected to obtain a network subproblem of the original problem (i.e. the H_g and R_a activities are modified so that they have only single inputs).

[FIGURE 8 ABOUT HERE]

In the Netform representation, activities are denoted by arcs while constraints are denoted by circles as before. The activities have associated upper and lower bounds (enclosed in parentheses), costs, and both head and tail multipliers. Unit values for multipliers and lower and upper bounds of (0,oo) on activity values are not shown explicitly. Networks with integer-valued activities (indicated by a # sign on the arc) are admissible. Omitting the non-network elements, Figure 8 is obtained simply from Figure 7. Multipliers at the heads of activities in the Netform representation correspond to activity output coefficients in the Resource/Activity diagram, while those at the tails correspond to activity input coefficients. Exogenous supplies and demands are shown as "dangling" arcs since they can be thought of as constant activities.

Experience with the Netform approach to modeling has been very positive, (Glover [1987]), confirming the value of graphical representations in the modeling process. A surprisingly large number of important integer and non-integer problems can be represented as networks. Many problems involving a time element, such as inventory and cash management applications, have a quite simple network representation. The rules for converting a Netform graph to an algebraic statement are straightforward being practically identical to those for an A-C diagram. In practical applications, side conditions, which do not adhere to the network restriction, may be present. These can be handled either by adopting the A-C representation for a part of the network, or by adding constraints/activities by hand to the algebraic statement of the network (see Glover [1987] for details).

Network diagrams which attempt to represent every activity and node are impractical for problems of even small to moderate size. Often, it is sufficient to develop a typical pattern of connections using a small number of graphical elements as an aid to writing down the equations in the problem statement. Figure 9 shows how the use of symbol names and set notation can simplify a Netform diagram and provide an excellent format for a computer interface.

[FIGURE 9 ABOUT HERE]

8. STRUCTURED MODELING

Structured Modeling (Geoffrion [1987]) represents a major effort towards building a sound basis for modeling theory and practice. Because of space limitations we can provide only a brief overview and illustration. The objective of structured modeling (SM) is to develop a comprehensive framework to unambiguously represent all the essential elements of a variety of management science models. This framework of definitions is to be represented in the computer and to be used to define and generate problem statements for the Solver, to test that a computable, consistent problem statement has been produced, to provide documentation for subsequent users of the model, to afford model-data and model-solver independence and to allow information about parts of the model and their relationships to be retrieved and displayed.

The elements in a structured model are as follows (from Geoffrion [1987]):

(1) Primitive Entity (PE): has no associated value and represents a thing or concept postulated as a primitive of the model (e.g. the "hens" element in the Farmer's problem).

(2) Compound Entity (CE): has no associated value and represents a thing or concept defined in terms of other things or concepts (e.g. a link between two locations in a transportation problem).

(3) Attribute (A): has a constant value and represents the value of a property of a thing or concept (e.g. the coefficients HAT_g, HLT, etc.).

(4) Variable Attribute (VA): similar to an Attribute except that its value is computed by the model (e.g. the variables HL, B_g, etc.).

(5) Function (F): has a value that can be computed from the other values in the model.(e.g. the term $\sum_g HAT_g.H_g$).

(6) Test (T): similar to a function but the result must be either true or false (e.g. a test to see if a constraint is satisfied).

SM models are specified in a rigorously defined syntax similar to a programming language. In a sense, SM can be thought of as a superset of an algebraic language, in which the data is defined by the PE's, CE's and A's and the algebraic statements by the VA's, F's and T's. These six model elements are related because (except for the primitive entities) each of the groups of elements ("genera") is defined in terms of elements from one or more of the preceding groups. This observation leads to the graph in Figure 10 in which the arcs (conventionally directed from PE's towards T's) can be interpreted as "the tail item is used in the definition of the head item". The "Genus" graph in Figure 10 is one of two principal types of graphs used in Structured Modeling. The other graph is a "Modular Tree" which depicts a hierarchical grouping of related element groups. A modular decomposition of the Farmer's Problem is indicated in Figure 10 but a Modular Tree is not shown.

[FIGURE 10 ABOUT HERE]

Roughly speaking, the relationship between the A-C graphs in Figure 7 and the Genus graph in Figure 10 is as follows:

(1) The Sets (including those with a single element) of Figure 7 are the PEs in Figure 10.

(2) The Coefficients are the Attributes (As).

(3) The Activity Nodes are the Variable Attributes (VAs).

(4) Each link from a Constraint node to an Activity node in Figure 7 represents a term in the LP. Terms (or summations of terms) can be represented by functions (F's) in Structured Modeling; these are shown as points in Figure 10.

(5) Each Constraint node in Figure 7 is replaced in Figure 10 by a Test node and one or more Function nodes (e.g. a Function node might gather together all the terms in a constraint to define its LHS).

The Genus graph for even a small problem is quite complicated to draw by hand. SM software should eventually generate these graphs automatically from the textual inputs. In terms of the concreteness dimension described earlier, SM problem statements and graphs are quite abstract; indeed, it is hard to discern the structure of the underlying problem in the graph of Figure 10. For these reasons, SM probably does not provide an ideal external representation for model specification. However, SM is obviously general across a wide range of modeling domains and techniques. In addition, it explicitly relates all the parts of a model in a consistent and complete fashion. It is therefore an excellent internal representation scheme and has been used in this way by Krishnan [1988]. This aspect of the SM model will be further elaborated in the next section.

9. DATABASE REPRESENTATION SCHEMES

The need to gather and process large quantities of data during the model building phase and to interpret the voluminous results obtained from large models, has prompted research directed towards the integration of modern database technology with mathematical programming systems (Dolk [1986] and [1988], Geoffrion [1987], Lenard [1987] and Choobineh and Sena [1988]).

There are two separate but related requirements. First, there is a need to record information about the structure of the model. Second, it is necessary to provide for the storage and manipulation of the data of the problem and of the results that are obtained from the optimizer. While model structure is probably handled best by data structures based on artificial intelligence techniques (Elam and Konsynski [1987]), the power of modern database management systems and query languages makes them attractive for the data manipulation aspects of modeling. In the following, a database approach (Date [1987]) will be used to illustrate the main issues for both requirements.

Figure 11 gives a conceptual view of the essentials of the graph in Figure 7 using the notation of the Entity-Relationship model (Chen [1977]). An E-R diagram depicts the things of interest to the system as entities (boxes) and relationships between entities (diamonds). Entities and relationships represent classes of objects whose individual instances are distinguished by the values of their associated attributes or properties.

[FIGURE 11 ABOUT HERE]

Figure 11 states that each Activity entity is related to one or more ("N") Constraint entities and each Constraint entity is related to one or more ("M") Activity entities. The Activity-Constraint relationship serves to relate the individual instances of the two entity sets and can carry information on the mathematical transformations linking each activity to each constraint. Also shown in Figure 11 are two entities used to record the results of the optimization for each activity and constraint.

Figure 12 gives a realization of this conceptual data model for the Farmer's Problem. We will call this the Model Schema.

[FIGURE 12 ABOUT HERE]

For a particular model, the model schema contains information that is useful in the following processing activities:

(1) Generation of the schema (skeletal outline) for the data tables that will store the data and results for the problem.

(2) Generation of both the algebraic representation of the problem and the MPS problem statement for input to the optimizer.

(3) Updating the model when structural relationships are changed.

The Activity, Constraint and Transform relations (data tables) in Figure 12 capture all the information in Figure 7. The Sets relation in the figure is redundant in the sense that it can be computed from the former three tables. However, it will obviously help speed processing.

The Model Schema in Figure 12 contains almost the same information as the Structured Modeling Genus Graph in Figure 10. The Sets relation in the Model Schema records the mappings between the PE's and the A's and VA's in the Genus Graph. The Activities, Constraints and Activities-Constraints relations record information concerning the F's and Tests in the SM representation. As shown in Murphy et al [1988], this is all the information needed to generate the algebraic form of the model in the case of LP's. To represent non-linear and other types of models, the Model Schema can be expanded, along the lines of the SM graph, to include an additional Function object; this would store the mathematical definition of the object. A desirable feature of the schema in Figure 12 as an internal representation, is that it contains information on the network structure underlying the model. To do this, it uses the information contained in the Upper- and Lower-bound and Input-Output fields.

Figure 13 shows a Data Schema and its instantiation with actual data values for the Farmer's Problem. Each set has been assigned a table of the same name to record set memberships. Similarly, each data coefficient has been assigned a database relation whose name is the name of the coefficient. The key (unique identifier for tuples in the relation) is the set of indices that describe the array position of the data coefficient in the LP matrix. Scalar objects have been treated as single element tables for uniformity of representation although they might be gathered together into a single table in an actual implementation.

[FIGURE 13 ABOUT HERE]

The Data Schema can be generated automatically from an analysis of the Model Schema (Asthana [1988]). The skeleton outlines for each Set table can be generated first and filled with element values, either interactively by the user, or automatically from knowledge stored previously in the system. Once the set memberships are known, it is possible to automate, or partially automate, the generation of the keys for the data elements in the coefficient tables. Finally, the data coefficient values can be filled-in, either automatically or by interaction with the user. It should be noted that data elements with unit values do not have to be stored if they can be implied from the algebraic statement.

The Data Schema in Figure 13 differs from the "Elemental Detail Tables" that are used for the same purpose in Structured Modeling (see Figure 14). In the former, each set and data element is represented by its own database table. In the latter, there is a data table for each Primitive and Compound Entity (i.e. for the sets); the coefficients are represented by database attributes and the elements of the sets by values in the same relation. It is difficult to decide between the two representations. The SM representation is much more compact, but the data schema in Figure 13 may be more flexible especially when data is to be shared between different models and modelers. Using the concept of database views (Date [1987]), it is possible to use one representation as the basis for the design of the physical database and to afford users the other view of the data depending on their tastes.

[FIGURE 14 ABOUT HERE]

From a relational database viewpoint, the matrices and higher dimensional arrays that are traditionally used by management scientists to represent the data of mathematical programs, are unnecessary. The relation for a coefficient stores only the non-zero elements in the array representation. Thus, it is a sparse representation that conforms closely to the MPS format used for input by most Optimizers. There is one table entry in Figure 13 for each non-zero entry in the LP matrix. Conceptually, all that is necessary to transform the database in Figure 13 into an MPS statement, is to replace the values of the keys in the relations by the appropriate (Row-label, Column-label) pairs. There is no need to generate arrays in the traditional sense unless the modeler prefers to view his/her model in this way.

In summary, the Model Schema is primarily an internal representation while the Data Schema is both an external and internal representation. Since the latter involves data rather than model structure, it can be used in conjunction with any of the other representation schemes discussed in the paper. Taking a different approach, Choobineh and

Sena [1988], suggest some extensions to the popular SQL database query language (Astrahan and Chamberlin [1988]) to support the expression of algebraic constraints. This has the advantage of providing a unified language for both the model definition and data manipulation phases of modeling. The disadvantages are as listed above for algebraic languages; the main drawbacks are that such languages are abstract rather than concrete and not as amenable to advanced interface support as the graphic representation schemes discussed earlier.

10. AN ICONIC REPRESENTATION SCHEME

The Activity-Constraint and Netform graphs are the most concrete (closest to the real world) representations reviewed so far. However, the nodes and arcs correspond directly to mathematical objects (the rows and columns of the model tableau) and only incidently to real world entities. The arguments in Section 2, and the success of "iconic" interfaces in many applications (Shneiderman [1987], Ch. 5), suggest the desirability of interfaces with more concrete images. Furthermore, even in their compact forms, the A-C and Netform graphs can be quite complicated implying the need for some form of hierarchical aggregation to simplify the problem for the user.
The LPGRAPH (Asthana [1988]) interface to the LPFORM system attempts to satisfy both of these goals. It has been implemented on an IBM PC/AT class machine using a set of graphics tools written in the "C" programming language (Expert Vision Associates [1988]).

The iconic representation of an LP problem in LPGRAPH consists of a hierarchy of networks which depict the problem in increasing detail. At each level in the hierarchy, the network consists of one or more "blocks" connected by directed arcs; blocks and arcs at lower levels in the hierarchy inherit properties from their parents at the next higher level. The blocks contain collections of zero or more LP activities. There are two kinds of directed arcs connecting the blocks. A "logical link" (shown by a thin line) indicates a flow that exists in the real world but is not modeled by an LP activity. An example is the flow of grains to animals in the farm problem, i.e. a material flow from one production point to another in a fixed sequence. A "flow link" (shown by a thick line) represents a flow that is modeled by an LP activity. A transportation activity is the commonest example. Icons are placed within the blocks to specify the existence of activities. In addition to a completely general activity icon, more specialized inventory and resource icons are provided for convenience. The idea of using activity icons during the formulation process first appears in Dantzig [1963].

We use the Farmer's Problem to illustrate the main ideas. The top level graph consists of a single "Farm-Problem" block. The representation at the second level of the hierarchy is shown in Figure 15.

[FIGURE 15 ABOUT HERE]

The operation of the farm is visualized as four separate functions (Administration, Crops, Husbandry and Marketing) each of which consists of a number of activities and is represented by a block on the diagram. Non-transportation links (logical links) between the blocks indicate the connections. As each activity icon is placed in its parent block, the user completes a fill-in-the-blank Activity screen. These are summarized in Figure 16. The user defines the activity index set and the input and output sets for each activity. To illustrate, HARVEST has "Grains" as its activity index set (i.e. there is a separate decision variable for each type of grain); its input sets are "Acres", "Labor" and "Dollars" (each of which is a singleton) and its output set is "Grains". As each input or output set is named, the system suggests a short name for the associated data coefficient according to the conventions in Asthana [1988]. These names are shown after the colons in Figure 16. They can be changed by the user (as has occurred for the unit coefficients in the figure).

[FIGURE 16 ABOUT HERE]

After the user has supplied the information in Figures 15 and 16, the Model Schema (Figure 12) and Data Schema (Figure 13 without the data values) are constructed internally. The algebraic statement (1) is generated and displayed using an algebraic language similar to that used by GAMS (see Section 3). The index matching rules provided in Murphy et al [1987] guarantee the completeness of the resulting model. Set memberships and the values of data coefficients must be specified at some point prior to running the model.

An entirely different strategy for defining the Farmer's Problem in LPFORM is to take a constraint- rather than an activity-oriented viewpoint. There are two ways of doing this. The first uses Constraint Screens that are, in a sense, the "duals" of the Activity screens outlined in Figure 16. Each constraint is defined in terms of the activities with which it interacts and the associated coefficient names. This approach avoids the use of mathematical notation by using the linearity property of LP's and certain relationships between index sets, to automatically generate the algebraic problem statements. The second method is to input the algebraic form of the problem statement directly using a language similar to that provided by GAMS. It is useful to combine the activity- and constraint-oriented approaches because actual applications often require that additional constraints be added to standard models defined from an activity perspective. Thus, a user might prefer to enter the ratio constraint (3) directly rather than by the method indicated in Figure 16.

Comparing Figure 7 and Figure 16 as alternative input representations for a computerized system, we see that only the activities have been defined in LPGRAPH; the user is not required to define either the Constraint Nodes nor the connections between the Activities and Constraints. This is an example of the "piecemeal" approach to problem specification mentioned earlier. Its advantage is that the user does less work (supplies the same information in less redundant form) and does not have to follow a rigid input sequence. The disadvantages are that users may feel uncomfortable about leaving things "up to the computer" and may not obtain as detailed an understanding about the way the model components relate. As mentioned earlier, preliminary results on the use of the graphics interface of LPGRAPH versus (its own) algebraic language are encouraging.

To illustrate some other features of iconic modeling, we use the following example:

"Warehouses purchase and store Raw-Materials prior to their transportation to Factories. The Factories maintain Raw-Materials and Products inventories. They use Raw-Materials to produce Products using a production process that has been modeled previously. Finally, Products are transported to Markets where they are sold."

The different types of entities and activities in the above problem are each represented, in a fairly obvious way, by an icon in Figure 17. Given this graph, the system requests the user to fill-in forms for the buy and sell activities, each inventory activity, each transportation flow and the production model. The input screens for the activities are used mainly to define their inputs and outputs (as described above for the Farmer's Problem). The input screen for the previously stored production model asks the user to match the names stored in the template model to the names for the same objects in the new model.

[FIGURE 17 ABOUT HERE]

The Flow, Inventory and Resource icons represent specialized kinds of activities and trigger user interactions which result in the addition of appropriate constraints to the model (see Ma [1988] for details). Resource icons are used to represent physical entities such as plant and equipment which are used by activities rather than consumed as with inventories. Other examples of LPGRAPH formulations are given in Ma [1988], Ma et al [1987 and 1989] and Asthana [1988].

Note that several, more complicated, graphs could be drawn to represent the above problem. First, one could draw a detailed transportation network showing individual warehouses, factories and markets together with all of the individual transportation routes. It is usually easier, however, to stop the drawing at the stage shown in Figure 17 and to let the detailed network connections be defined through the data. As a second alternative, one could draw an A-C graph using the conventions in Figure 7. However, this graph would be quite complicated as the model involves flows in both space and time. In effect, the LPGRAPH system automatically recognizes the network substructure of the problem and implicitly makes the connections of the underlying A-C graph as it generates the algebraic representation. The detailed connections of the transportation network are obtained from the data when the MPS format of the problem is generated.

The above paragraphs illustrate several features which should help the modeler. These include a simple, non-mathematical representation, hierarchical problem definition (only the top-most graph was drawn in this instance), bottom-up construction of the model (use of the previously developed production model) and a piecemeal approach to problem definition (it was not necessary to adhere to a rigid order in defining the problem to the computer nor even to supply all the detail concerning interrelationships between model elements). Users are however, required to maintain consistent naming conventions so that the system can sort and assemble the components of the problem (see Murphy et al [1987], for a detailed description of how the model components can be generated and assembled).

An LPFORM graph (c.f. Figure 15) can be viewed as an aggregated form of A-C graph (c.f. Figure 7). An A-C graph can be simulated in LPFORM by making the following correspondences: use LPFORM blocks containing a single activity to represent A-C Activity nodes, blocks containing no activities to represent A-C Constraint nodes, and "logical" flows connecting the appropriate blocks to represent the arcs in the A-C graph. Note that the data coefficients appear with the activities in LPFORM rather than on the arcs as in an A-C graph.

If there are no submodel icons, the steps to transform an LPFORM graph into an equivalent compact A-C graph are as follows:

(1) For each LPFORM Activity, Inventory or Resource icon, attach a node to the tail end of each of its input arcs and to the head of each of its output arcs. In the A-C diagram, these will represent constraints on the inputs and outputs of resources to activities. Replace the LPFORM activity icon by its open box representation in the A-C graph.

(2) For each LPFORM block, add its index sets to the index sets of its activities and to the index sets of the resource nodes constructed in step (1). Discard the block icon.

(3) Replace each LPFORM flow activity (arc) by an A-C activity icon connected to the appropriate output resource node in the block at the tail of the LPFORM arc and the appropriate input resource node in the block at the head of the LPFORM arc.

(4) Complete the A-C graph by replacing multiple instances of the same constraint nodes by single instances while maintaining all connections.

Thus, we can translate from one representation scheme to another except that we lose information on the hierarchical structure if we go from the iconic representation to the A-C diagram and back.

In the special case where the model is a pure or generalized network problem, there is an LPFORM graph which is equivalent to the Netform graph for the problem. In this graph, the LPFORM blocks and flows correspond, respectively, to the NETFORM resource nodes and arcs. The major difference between the two graphs is that exogenous supplies and demands are shown as blocks in LPFORM rather than as dangling arcs as in Netform.

To summarize, iconic representation schemes can provide an elegant method for specifying large LP's. The ability to define the problem piece-wise and in non-algebraic terms should also be helpful to both experts and nonexperts. In any case, we believe that it is fruitful to provide a number of different, interchangeable, representation schemes within a common framework. The LPGRAPH interface therefore combines elements from the Activity-Constraint graph, Netform, database and algebraic representation schemes. Preliminary results from an experiment which directly compared groups of users formulating LP problems using the graphical and algebraic languages provided by LPFORM, show that the former group obtained a higher percentage of correct solutions in a shorter time and were more satisfied with their experience (Asthana [1988]). The relative advantage of the graphics package increased with the complexity of the problem.

11. CONCLUSIONS

The microcomputer revolution has increased computer literacy and familiarity with models (at least of the spreadsheet variety). The current proliferation of powerful desk-top workstations in all forms of office and professional work provides a tremendous opportunity for management science. The algorithms and analytic techniques developed over the last forty years can now influence policy makers in a much broader array of applications and situations. The challenge is to develop software environments that will improve the productivity of the modeling process, the quality of the models produced and, most importantly, the quality of the decisions based on the use of these models.

This paper has reviewed some methods for representing mathematical problems in graphical and/or textual formats most of which avoid the use of algebra or other essentially mathematical representations. As we have tried to show, the representations are largely equivalent in that transformations exist from one form into another. We believe however, that they differ in terms of the amount of work, skill and understanding that is required from users. Thus, we need both external and internal representations that take into account the cognitive limits of human beings. Because of the importance we attach to this issue, the paper has introduced "compact" forms of both A-C graphs and Netforms. The iconic representations in the previous section go one step further in the sense that they are aggregated forms of the A-C graphs.

In Section 2, we proposed four dimensions for characterizing external representation schemes: generality (the applicability of the technique to a range of management science models), concreteness (how closely real world objects are represented), labor-intensiveness (a function of the complexity of the representation) and interface potential (how well the representation lends itself to advanced computer interface techniques). From the discussion in this paper it is apparent that no representation scheme dominates the others on all of these dimensions. In fact, we believe that an optimal modeling system will employ more than one form of external model representation. (Naturally, only one form of internal representation is desirable).

Actual experience with the representation schemes discussed in this paper in real work environments will be necessary before their usefulness in promoting more effective use of modeling in organizations can be properly evaluated. In the final analysis, the choice of a particular representation scheme will depend on the circumstances and on the tastes of users as each method has its advantages and disadvantages. It is possible to build systems that avoid unfortunate trade-offs between user convenience, generality of representation and machine efficiency. Thus, one can have mixed representations at the user interface that allow iconic, network and algebraic techniques to be used to define different parts of the same model. The resulting external specification can then be translated automatically into an unambiguous and valid statement that is stored and analyzed internally using, for example, the techniques of Structured Modeling. Note that the iconic representation and Structured Modeling have a natural fit in their hierarchical structuring of problems since a block in the former is equivalent to a module in the latter.

Much research remains to be done in the area of modeling interfaces. In our view, the greatest problem facing designers of languages to support the modeling process involves the trade-off between the need to present a precise, unambiguous input to the optimizer and the limited cognitive capabilities of human beings. There is a great need for improvement in our understanding of the issues in this area. For example, we have proposed:

(1) Multiple methods for representing problems and parts of the same problem.

(2) Graphic representation schemes including the use of icons.

(3) Dynamic support for problem solving strategies such as hierarchical decomposition and piece-wise composition.

The effectiveness of the above interface features is a matter for research. Certainly, no prima facie argument can prove their desirability. For example, multiple representation schemes may be confusing to users. Furthermore, the domain in which graphical representations are natural and easy to understand without training in operations research, is somewhat limited. In some applications, the entities represented by such graphs are far from concrete in the sense that we have been using the term. For example, in a network representing a cash flow application, the nodes represent time periods and the arcs represent cash flows both within a period and between periods. Thus, there is a need to develop and experiment with graphical representations for more abstract entities and, in particular, to extend the techniques to include nonlinear and integer programming. Finally, we need a better understanding of the problem solving strategies of expert modelers so that we can build systems that can truly extend their capabilities.

The power of modern computers, their graphic capabilities, new forms of man-machine communication (other than typing at a keyboard), and the emergence of artificial intelligence techniques, all point to an exciting period of research and development which will result in modeling workstations of tremendous power and versatility. New problem representation techniques will play an essential role in this evolution and represent an important new area of management science research.

Acknowledgements:

The authors wish to thank the referees and Dan Dolk and Arthur Geoffrion for their helpful suggestions. This research was supported by the Amoco and Shell oil companies.

REFERENCES

Asthana, A., "LPGRAPH: An Expert System for Graphically Formulating Linear Programs", Ph.D. Thesis, New York University (1988).

Astrahan, M. M., and D. D. Chamberlin, "Implementation of a Structured Programming Language", **Communications of the ACM**, Vol. 18, No. 10, (October 1988), pp. 580-588.

Baker, T. E., "Integrating AI/OR/Database Technology for Production Planning and Scgeduling", Technical Report, Chesapeake Decision Sciences, New Providence, New Jersey (1990).

Binbasioglu, M. and Jarke, M., Domain Specific Tools for Knowledge-Based Modeling", **Decision Support Systems**, Vol. 2 (1986).

Blanning, R. W., "A Relational Framework for Model Management in Decision Support Systems", **DSS-82 Transactions** (1982), pp. 16-28.

Bodily, S., "Spreadsheet Modeling as a Stepping Stone", Interfaces, Vol. 16, No. 5 (1986), pp. 34-52.

Bradley, Gordon H. and Robert D. Clemence, Jnr., "A Type Calculus for Executable Modeling Languages", **IMA Journal of Mathematics and Management**, Vol. 1, 1987, pp. 277-291.

Breitman, R. L. and J. M. Lucas, "PLANETS: a Modeling System for Business Planning", **Interfaces**, Vol. 17, No. 1 (Jan-Feb 1987).

Brown, R. W., W. D. Northup and J. F. Shapiro, "LOGS: A Modeling and Optimization System for Resource Planning". In: **Computer Methods to Assist Decision Making**, New York, North-Holland (1987).

Carlson, E., B. Grace and J. Sutton, "Case Studies of End User Requirements for Interactive Problem Solving", **MIS Quarterly**, Vol 1, No 1, (March 1977).

Chen, P. P. S., "The Entity-Relationship Model: Towards a Unified View of Data", **ACM Transactions on Database Systems**, Vol. 1, No 1., (March 1976), pp. 9-36.

Choobineh, Joobin, and James A. Sena, "A Data Sublanguage for Formulation of Linear Mathematical Models", **Proc. Twenty First Annual Hawaii Conference on System Sciences**, Kona, Hawaii (1988), pp. 340-348.

Creegan, J. B. Jr., "Dataform: A Model Management System", Ketron Management Science, Inc., Arlington, Virginia, November (1985).

Dantzig, George B., **Linear Programming and Extensions**, Princeton University Press, Princeton, N.J. (1963).

Date, C. J., **An Introduction to Database Systems**, Addison-Wesley, Reading, Mass. (1987).

Dolk, D., "A Generalized Model Management System for Mathematical Programming", **ACM Transactions on Mathematical Software**, Vol. 12, No. 2 (June 1986), pp. 619-628.

Dolk, D., "Model Management and Structured Modeling: The Role of an Information Resource Dictionary System", **Communications of the ACM**, Vol. 31, No. 6 (June 1988), pp.704-718.

Egli, G. H. and J. Kohlas, "A Policy Model for Planning Alimentary Self-Sufficiency in Switzerland", **J. of Operational Research**, pp. 312-322, (1981).

Elam, J. J. and B. Konsynski, "Using Artificial Intelligence to Enhance the Capabilities of Model Management Systems", **Decision Sciences**, Vol. 18 (1987), pp. 487-502.

Expert Vision Associates, "EVA UIMS/GDB, User Manual", Cupertino, California (1988).

Fabozzi, J. F., and J. Valente, "Mathematical Programming in American Companies: A Survey", **Interfaces**, Vol. 7, No. 1 (November 1976).

Fourer, R., D. M. Gay and B. W. Kernighan, "A Modeling Language for Mathematical Programming", **Management Science**, Vol. 36, No. 5, pp.519-544 (1990).

Geoffrion, A. M., "Introduction to Structured Modeling", **Management Science**, Vol. 33, No. 5 (1987), pp. 547-588.

Glover, Fred, "Notes on Netforms", University of Colorado, Boulder (1987).

Glover, Fred, John Hultz and Darwin Klingman, "Improved Computer-Based Planning Techniques", Part I, **Interfaces**, Vol. 8, No. 4, pp. 16-25, and Part II, **Interfaces**, Vol 9, No. 4, pp 12-20 (1978).

Greenberg, H. J., "A Functional Description of ANALYZE: A Computer-Assisted Analysis System for Linear Programming Models", **ACM Transactions on Mathematical Software**, Vol. 9, No. 1 (1983), pp. 18-56.

Greenberg, H. J., "A Natural Language Discourse Model to Explain Linear Programming Models and Solutions", **Decision Support Systems**, Vol. 3, (1987), pp. 333-342.

Haverly Systems Inc., "OMNI Linear Programming System: User Manual and Operating Manual", Denville, N.J. (1977).

Hurlimann, Tony, **LPL: A Structured Language for Modeling Linear Programs**, Verlag Peter Lang AG, Bern (1987), Ch. 4.

IBM Corporation, "IBM Mathematical Programming Language Extended/370 (MPSX/370)", Program Reference Manual, SH19-1095, Paris, France (1975).

Klingman, Darwin, Fred Glover and Nancy Phillips, "Netform Modeling and Applications", Working Paper, Center for Business Decision Analysis, The University of Texas at Austin, (June 1989).

Krishnan, R., "Knowledge-Based Aids for Model Construction", Ph.D. Thesis, University of Texas, Austin (1987).

Lenard M., "An Object-Oriented Approach to Model Management", **Proc. Twentieth Hawaii International Conference on System Sciences**, Vol I, January 1987, pp. 509-515.

Lucas C. and G. Mitra, "CAMPS: Preliminary User Manual", Dept. of Mathematics and Statistics, Brunel University, U.K. (July 1985).

Ma, Pai-chun, "An Intelligent Approach Towards Formulating Linear Programs", Ph.D. Thesis, New York University (1988).

Ma, P., F. H. Murphy and E. A. Stohr, "A Graphics Interface for Linear Programming", **Communications of the ACM**, Vol 32, No 8 (August 1989), pp. 996-1012.

Ma, P., F. H. Murphy and E. A. Stohr, "Computer-Assisted Formulation of Linear Programs", **IMA Journal of Mathematics in Management**, Vol 2, September (1987).

Meeraus, A., "General Algebraic Modeling System (GAMS): User's Guide, Version 1.0", Development Research Center, World Bank (1984).

Murphy, F. H. and E. A. Stohr, "An Intelligent System for Formulating Linear Programs", **Decision Support Systems**, Vol. 2, No. 1 (Jan-Feb 1986).

Murphy, F. H., E. A. Stohr and P. Ma, "Composition Rules for Building Linear Programs from Component Models", Working Paper No. 148, Center for Research in Information Systems, Graduate School of Business Administration, New York University, New York (1987).

Orlikowski, W. and V. Dhar, "Imposing Structure on Linear Programming Problems: an Empirical Analysis of Expert and Novice Models", **Proc. National Conference on Artificial Intelligence**, Philadelphia, Pennsylvania (August 1986).

Raghunathan, Srinivasan, "An Intelligent Decision Support System for Model Formulation", Working Paper, University of Pittsburgh (1987).

Sander, R. L. and M.G. Smith, "A Description of Bonner & Moore's GAMMA System", No. CSH-007, Bonner & Moore Software Systems, Houston, Texas, (1976).

Shneiderman, B., **Designing the User Interface: Strategies for Effective Human-Computer Interaction**, Addison-Wesley, Reading, Mass. (1987).

Schrage, Linus, **Linear, Integer and Quadratic Programming with LINDO**, Scientific Press, Palo Alto, (1987).

Shepherd, R. N., "Recognition Memory for Words, Sentences and Pictures", **J. of Verbal Learning and Verbal Behavior**, Vol. 6, No. 2, (February 1987), pp. 156-163.

Welch, James S., Jr., "PAM: A Practitioner's Guide to Modeling", **Management Science**, Vol. 33, No. 5 (May 1987), pp. 610-625.

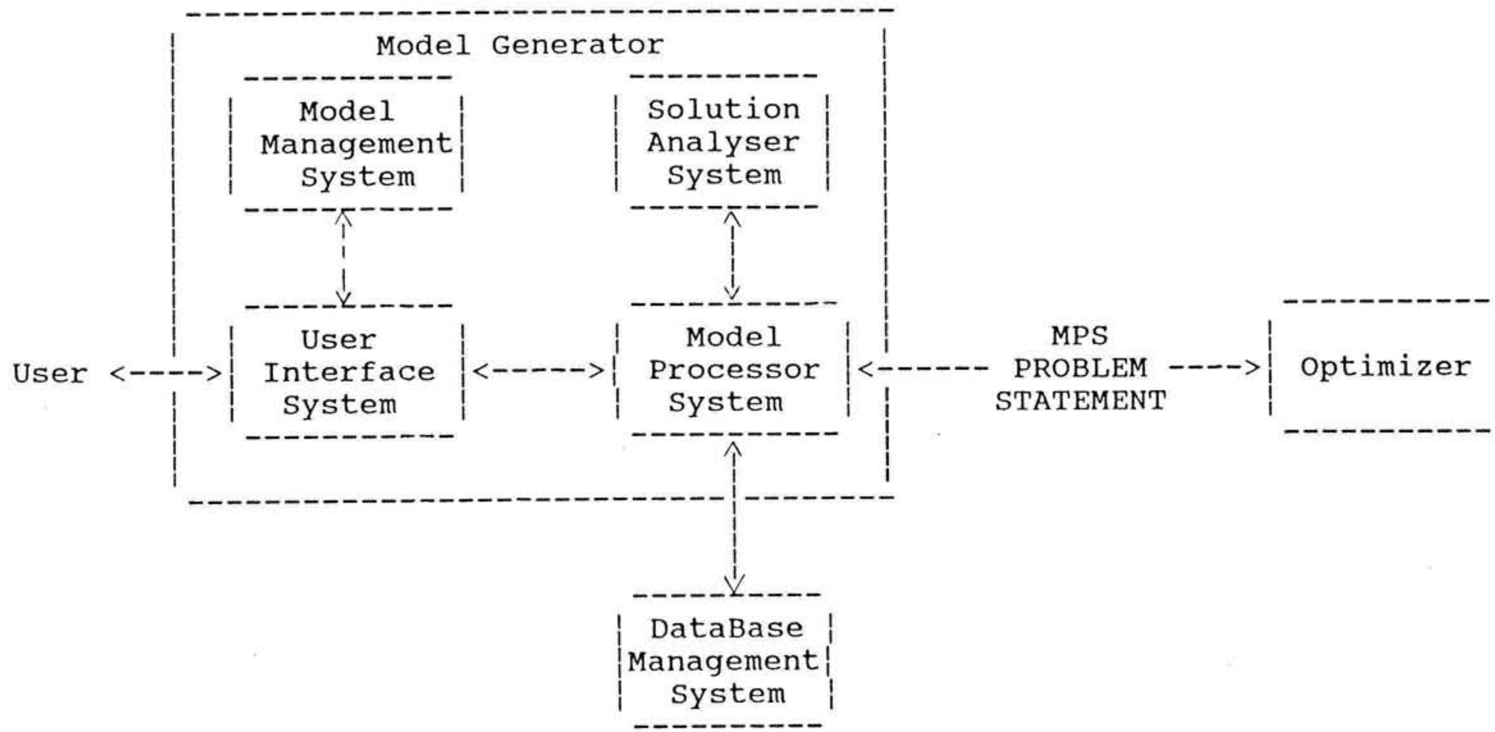


Figure 1
ARCHITECTURE OF LP MODELING SYSTEMS

	BUY-GRAINS		HIRE-LABOR	HARVEST-GRAINS		FEED-ANIMALS-GRAINS				SELL-GRAINS		RAISE-ANIMALS			
	Bw	Bc	HL	Hw	Hc	Fhw	Fhc	Fpw	Fpc	Sw	Sc	Rh	Rp		RHS
ACRES				1/55	1/95									≤	120
LABOR			0.85	-0.22	-0.74							-40	-25	≤	4000
WHEAT	1			1		-1	-1			-1				=	0
CORN		1			1			-1	-1		-1			=	0
HEN-FEED						0.10		0.04				-1		=	0
PIG-FEED							0.04		0.05				-1	=	0
FLOOR												15	25	≤	10000
HIRE-BOUND			1											≤	13333
PROFITS	-2.50	-1.50	-1.50	-0.20	-0.12					1.75	0.95	40	40		

Figure 2
LP TABLEAU FOR FARMER'S PROBLEM

VARIABLES:

LONG NAME	SHORT NAME	DESCRIPTION
BUY	B(g)	Buy grain g (bushels)
HIRE-LABOR	HL	Hire labor (hours)
HARVEST	H(g)	Harvest grain g (bushels)
FEED	F(a,g)	Feed animal a grain g (bushels)
SELL	S(g)	Sell grain g (bushels)
RAISE	R(a)	Raise animals a (units)

OBJECTIVE COEFFICIENTS:

NAME	VALUE	DESCRIPTION
HLC	1.50	Hire-labor Cost ($/hour)
BC(g)	2.50 1.50	Buy Grains Cost ($/bushel)
HC(g)	0.20 0.12	Harvest Cost ($/bushel)
SP(g)	1.75 0.95	Selling Price ($/bushel)
RP(a)	40 40	Raise Price ($/animal)

CONSTRAINTS:

LONG NAME	SHORT NAME	DESCRIPTION
ACRES-USAGE	AU	Acres balance equation (acres)
LABOR-USAGE	LU	Labor usage (hours)
GRAIN-BALANCE	GB(g)	Grains balance equations (bushels)
ANIMALS-BALANCE	AB(a)	Animals balance equations (units)
FLOOR-USAGE	FU	Floor usage (sq. ft.)
HIRE-BOUND	BND	Bound on hired labor (hours)

RHS COEFFICIENTS:

NAME	VALUE	DESCRIPTION
AS	120	Acres supply (acres)
HLS	13,333	Hire-Labor supply ((Hours)
LS	4,000	Labor supply (hours)
FS	10,000	Floor supply (sq. ft.)

TECHNOLOGY COEFFICIENTS:

NAME	VALUE	DESCRIPTION
HLLT	0.85	Hire-Labor/Labor Technology (hours/hour)
HAT(g)	1/55 1/95	Harvest/Acres Technology (acres/bushel)
HLT(g)	0.22 0.74	Harvest/Labor Technology (bushels/hour)
[illegible](a,g)	|0.1 0.04| |0.04 0.05|	Feed/Animals Technology (animals/bushel
[illegible](a)	40 25	Raise/Labor Technology (hours/animal)
[illegible](a)	15 25	Raise/Floor Technology (sq. ft./animal)

SETS:

LONG NAME	INDEX NAME	MEMBERSHIP
Grains	g	{wheat, corn}
Animals	a	{hens, pigs}

Figure 3
DATA DEFINITIONS FOR FARMER'S PROBLEM

DATA Z:COLUMNS COLUMN DESCRIPTION

**	BUY	HIRE	HARVEST	FEED	SELL	RAISE
L	B	HL	H	F	S	R
1				ANIMALS		ANIMALS
2	GRAINS		GRAINS		GRAINS	

DATA Z:MATRIX MATRIX SCHEMATIC

**	BUY	HIRE	HARVEST	FEED	SELL	RAISE	RTYPE	RHS
PROFIT	BC	HLC	HC		SP	RP	<>	
AU			HAT				<	AS
LU		HLLT	-HLT			-RLT	<	LS
GB	/1		/1	/-1	/-1		=	
AB			FAT			/-1	=	
FU					RFT		<	FS
BND		/1					UB	HLS

DATA Z:ROWS ROW DESCRIPTION

**	L	1	2
ACRES-USE	AL		
LABOR-USE	LU		
GRAIN-BAL	GB		GRAINS
ANIMAL-BAL	AB	ANIMALS	
FLOOR-USE	FU		

DATA Z:DATA DATA TABLE DESCRIPTIONS

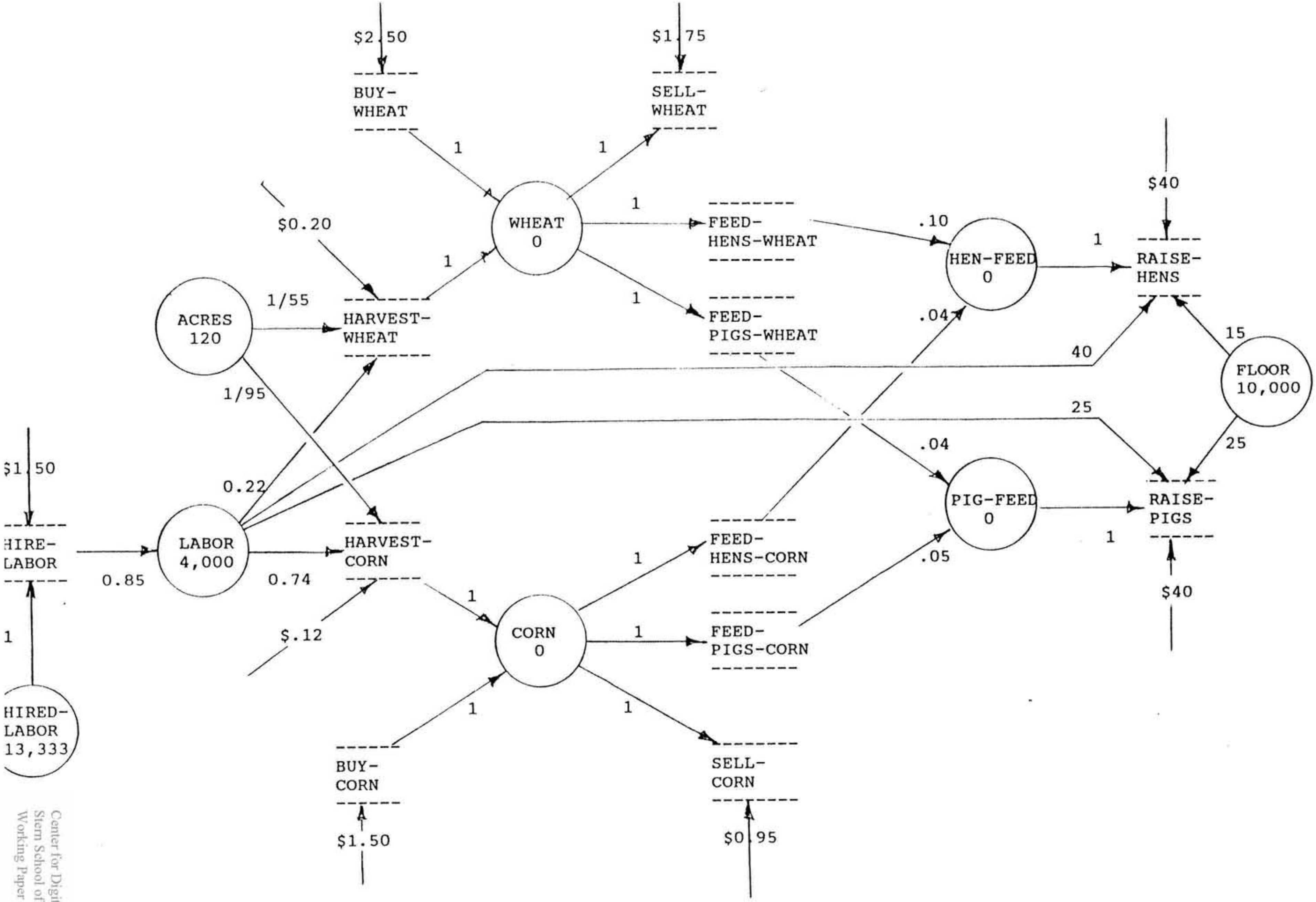
DATA Z:DATA DATA TABLE DESCRIPTIONS

**	TABLE	STUB	HEAD
HAT	FARM	GRAINS	HAT
HLT	FARM	GRAINS	HLT
FAT	FAT	ANIMALS	GRAINS
RLT	FARM	GRAINS	RLT
RFT	FARM	GRAINS	RFT
"	"	"	"
"	"	"	"

Figure 4
BLOCK-SCHEMATIC VIEW OF FARMER'S PROBLEM

```
        SETS
            G       grain    /WHEAT, CORN/
            A       animals  /HENS,PIGS/ ;
        SCALARS
            HLLT    effective hours per hired hour /.85/
            LS      starting quantity of labor     /4000/
            AS      acreage on farm                /120/
            FS      floor space                    /10000/
            HLC     wage rate per hour             /1.50/
            HLS     limit on hired labor           /13333/ ;
        PARAMETERS
            BC(G)   buy grains cost            /WHEAT  2.50
                                                CORN   1.50/
            HC(G)   harvest grains cost        /WHEAT  .20
                                                CORN   .12/
            SP(G)   sell grains price          /WHEAT  1.75
                                                CORN   .95 /
            RP(A)   raised animals price       /HENS   40
                                                PIGS   40/
            HAT(G)  harvest acres per bushel   /WHEAT  .01818
                                                CORN   .01053/
            HLT(G)  harvest labor per bushel   /WHEAT  .223
                                                CORN   .742/
            RLT(A)  raise animals labor        /HENS   40
                                                PIGS   25/
            RFT(A)  raise animals floor        /HENS   15
                                                PIGS   25/ ;
        TABLE
            FAT(A,G) feed animals technology
                   WHEAT   CORN
            HENS   .1       .04
            PIGS   .04      .05  ;
        VARIABLES
            B(G)    buy grains
            HL      hire labor
            H(G)    harvest grain
            F(G,A)  feed grain to animals
            S(G)    sell grain
            R(A)    raise animals
            Z                ;
        POSITIVE VARIABLES  B, HL, H, F, S, R ;
            HL.UP = HLS ;
        EQUATIONS
            PROFIT
            AU      acres
            LU      labor
            GB(G)   grain balance
            AB(A)   animal balance
            FU      floor usage   ;

        PROFIT.. Z =E=  -SUM(G,BC(G)*B(G)) -HLC*HL
               -SUM(G,HC(G)*H(G))
                        +SUM(G,SP(G)*S(G)) +SUM(A,RP(A)*R(A)) ;

        AU..   SUM(G,HAT(G)*H(G))  =L= AS            ;

        LU..     -HLLT*HL +SUM(G,HLT(G)*H(G))
                              +SUM(A,RLT(A)*R(A)) =L= LS;

        GB(G)..  -B(G) -H(G) +SUM(A,F(G,A)) +S(G) =E= 0    ;

        AB(A)..  SUM(G,FAT(A,G)*F(G,A)) - R(A) =E= 0    ;

        FU..     SUM(A,RFT(A)*R(A)) =L= FS ;

MODEL FARM /ALL/ ;
SOLVE FARM USING LP MAXIMIZING Z ;
```

Figure 5 .
FORMULATION OF FARMERS PROBLEM IN GAMS



The node for the dollar resource has been omitted.

Figure 6
ACTIVITY-CONSTRAINT GRAPHICAL REPRESENTATION

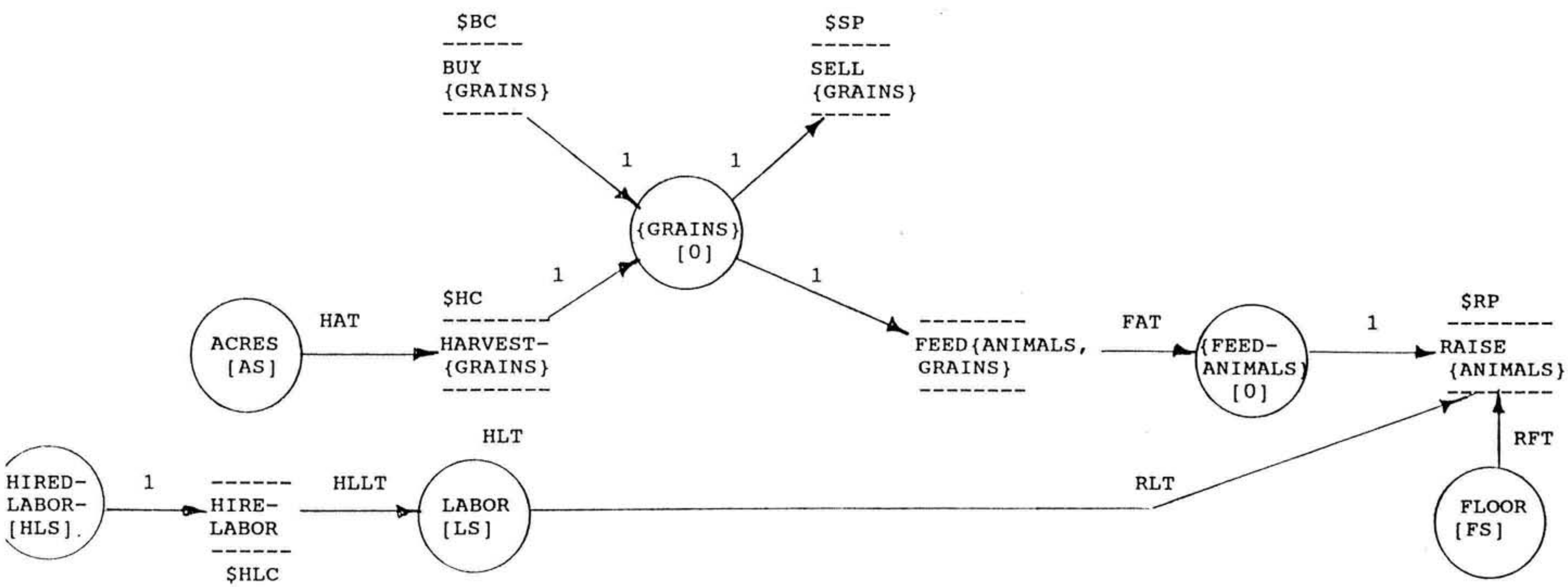


Figure 7A
COMPACT ACTIVITY-CONSTRAINT REPRESENTATION

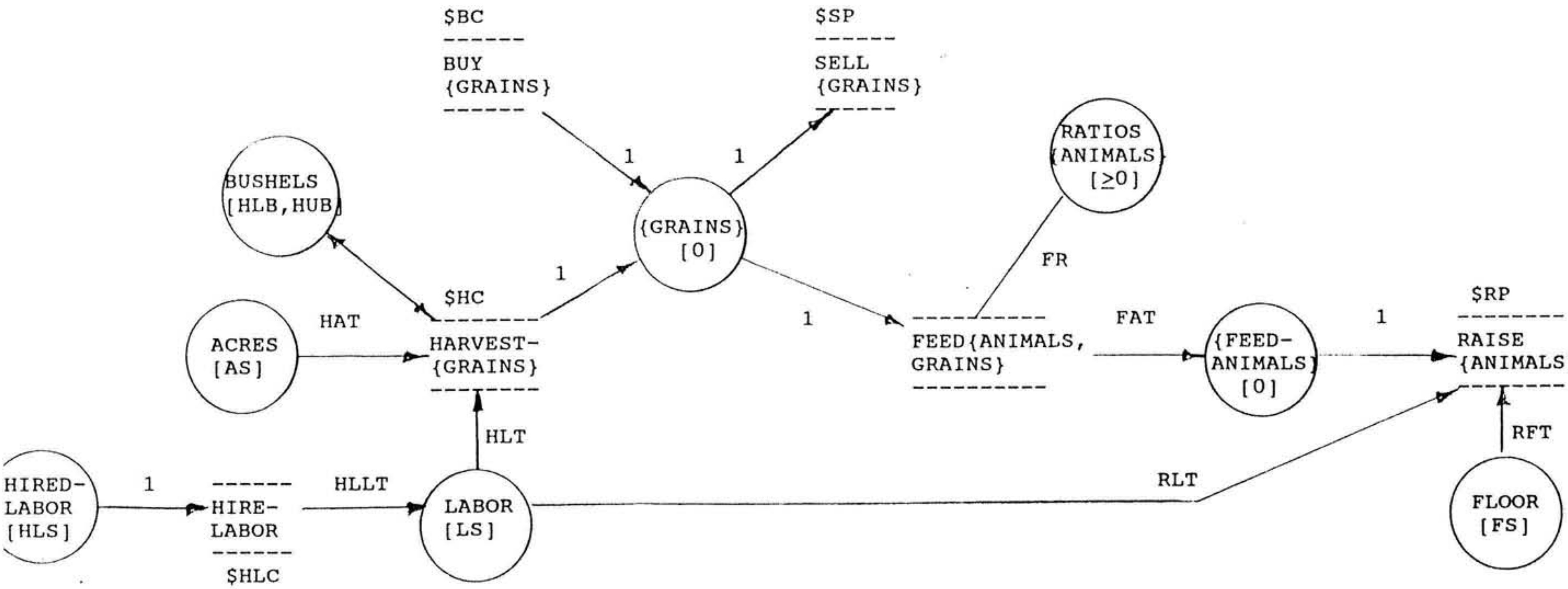


Figure 7B
AC-GRAPH WITH SIDE CONSTRAINTS

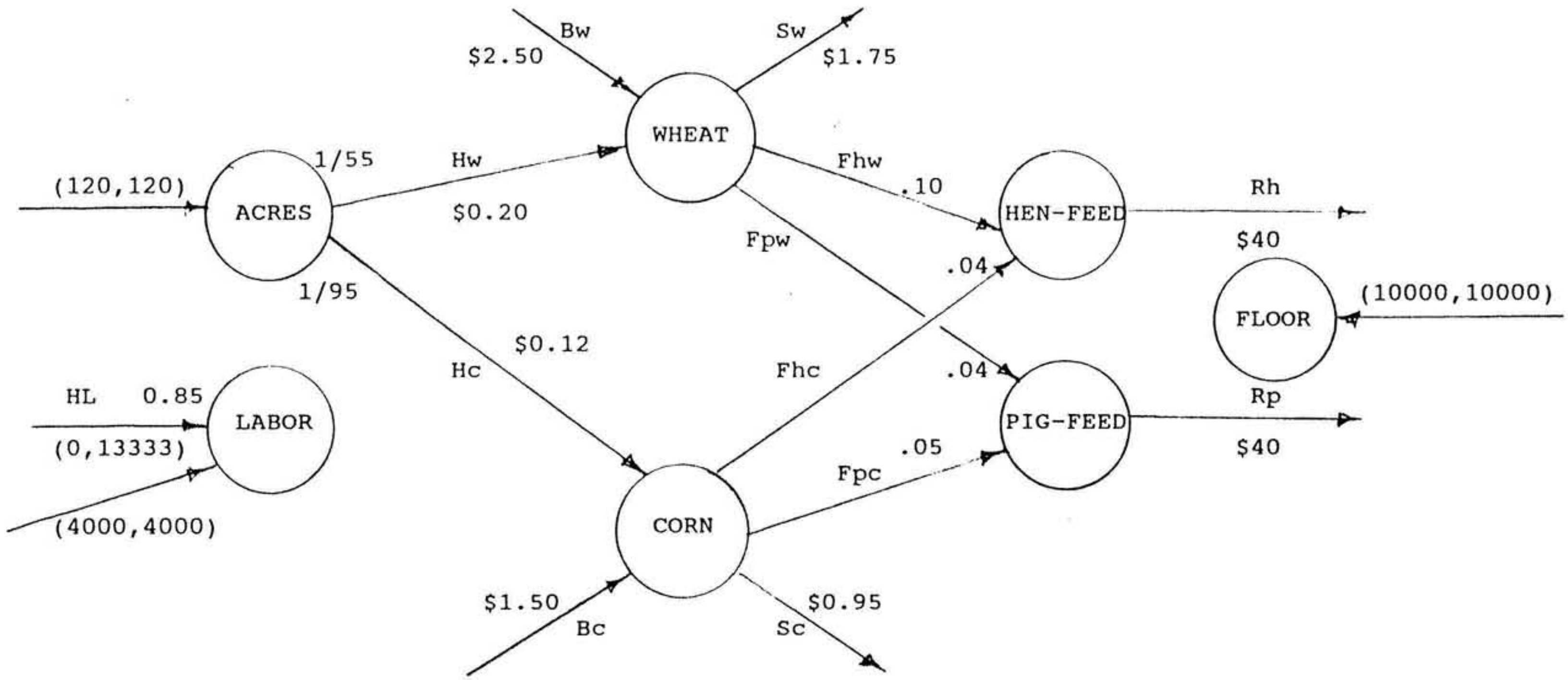
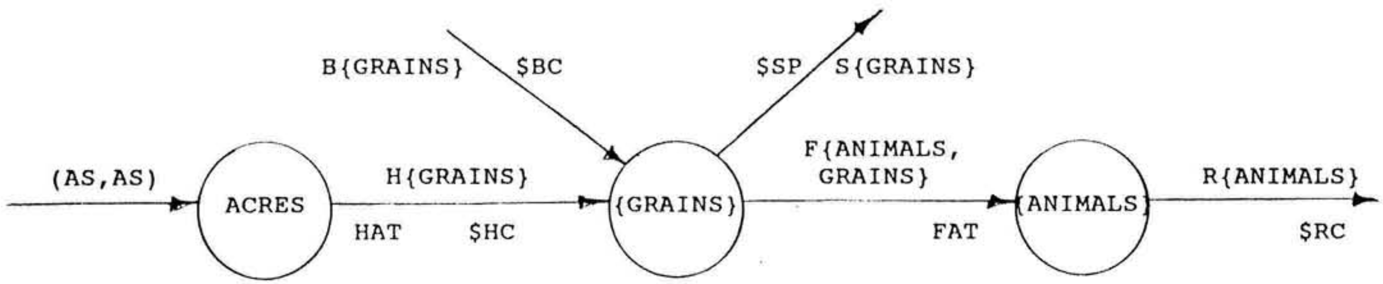
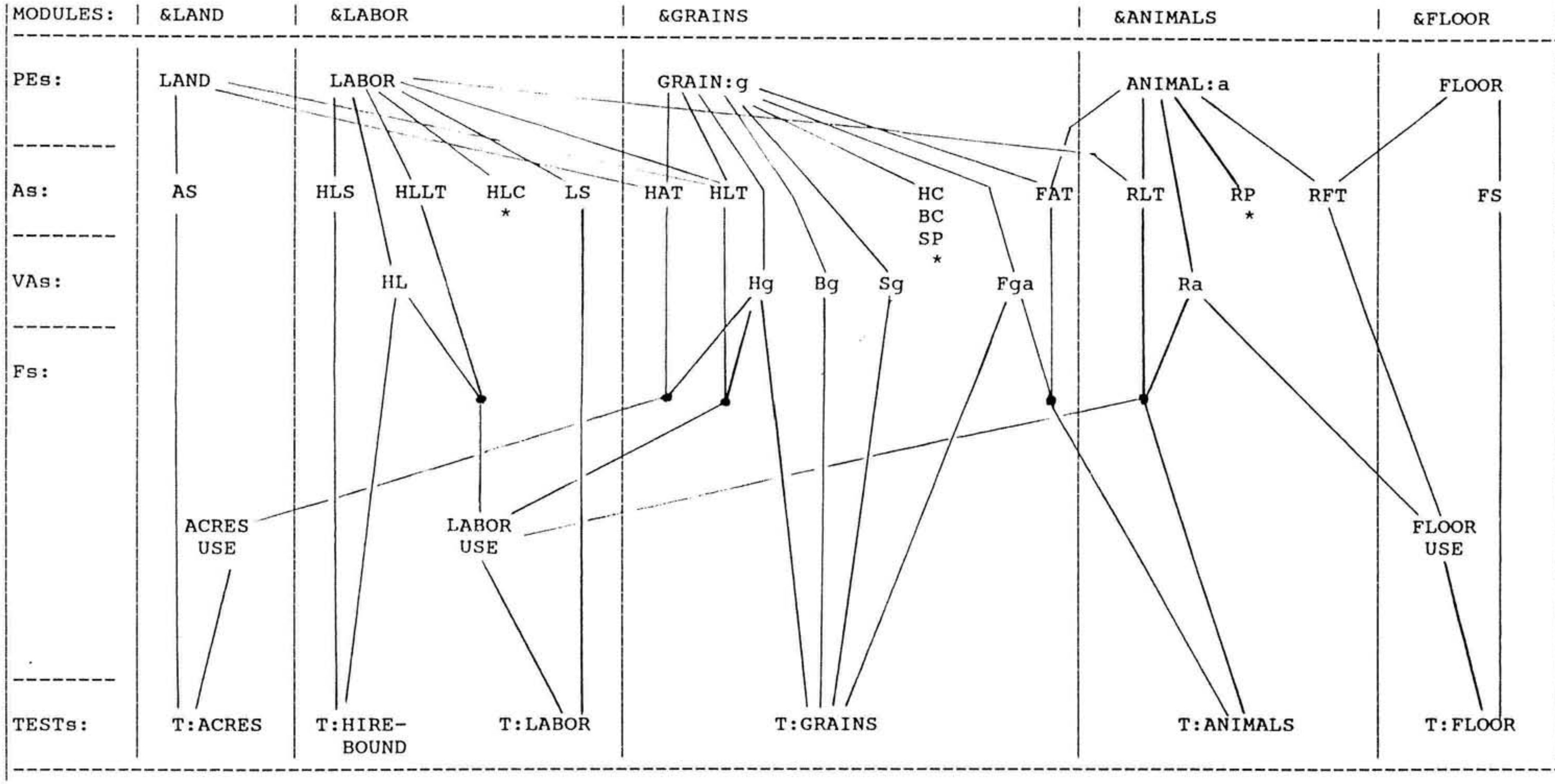


Figure 8
NETFORM GRAPHICAL REPRESENTATION



Note: The graph shows a network subproblem of the original problem.

Figure 9
COMPACT NETFORM REPRESENTATION



* Note: Connections for objective function not shown.

Figure 10
STRUCTURED MODELING GENUS GRAPH

Figure 11
ENTITY-RELATIONSHIP DIAGRAM: MODEL SCHEMA

ACTIVITY:

SHORT NAME	LONG NAME	OBJECT-COEFFT	ACTIVITY-SET	A-LOWER BOUND	A-UPPER BOUND
B(g)	BUY	BC(g)	GRAINS	0	Inf
HL	HIRE-LABOR	HLC	Nil	0	HLS
H(g)	HARVEST	HC(g)	GRAINS	0	Inf
F(a,g)	FEED	1	ANIMALS-GRAINS	0	Inf
S(g)	SELL	SP(g)	GRAINS	0	Inf
R(a)	RAISE	RP(a)	ANIMALS	0	Inf

ACTIVITY-RESULT:

SHORT NAME	OPTIMAL-VALUE	REDUCED-COST	A-LOWER-RANGE	A-UPPER-RANGE
B(hens)				
B(pigs)				
HL				
H(wheat)				
H(corn)				
"				
"				

CONSTRAINT:

SHORT NAME	LONG NAME	CONSTRAINT-SET	C-LOWER-BOUND	C-UPPER-BOUND
AU	ACRES-USAGE	Nil	0	HLS
LU	LABOR-USAGE	Nil	0	0
GB(g)	GRAINS-BALANCE	GRAINS	0	0
AB(a)	ANIMALS-BALANCE	ANIMALS	0	LS
FU	FLOOR-USAGE	Nil	0	0
Z	PROFITS	Nil	-Inf	Inf

CONSTRAINT-RESULT:

SHORT NAME	SLACK-VALUE	DUAL-VALUE	C-LOWER-RANGE	C-UPPER-RANGE
AU				
LU				
GB(hens)				
GB(pigs)				
"				
"				
"				

TRANSFORM:

CONSTRAINT	ACTIVITY	TRANS-COEFFT	INPUT-OUTPUT
AU	H(g)	HAT(g)	I
LU	HL	HLLT	O
LU	H(g)	HLT(g)	I
LU	R(a)	RLT(a)	I
GB(g)	H(g)	1	O
GB(g)	B(g)	1	O
GB(g)	F(a,g)	1	I
GB(g)	S(g)	1	I
AB(a)	F(a,g)	FAT(a,g)	O
AB(a)	R(a)	1	I
FU	R(a)	1	I

SETS:

SHORT NAME	USED-BY	USE-TYPE
GRAINS	B(g)	Activity
GRAINS	S(g)	Activity
GRAINS	H(g)	Activity
GRAINS	F(a,g)	Activity
GRAINS	GB(g)	Constraint
GRAINS	HAT(g)	Coefft
GRAINS	HLT(g)	Coefft
GRAINS	FAT(a,g)	Coefft
GRAINS	BC(g)	Coefft
GRAINS	HC(g)	Coefft
GRAINS	SP(g)	Coefft
GRAINS	Animals-Grains	Set
ANIMALS	F(a,g)	Activity
ANIMALS	R(a)	Activity
" "	" "	" "
" "	" "	" "

Figure 12
MODEL SCHEMA FOR FARMER'S PROBLEM

SETS:

GRAINS

GRAINS
Wheat
Corn

ANIMALS

ANIMALS
Hens
Pigs

ANIMALS-GRAINS

ANIMALS	GRAINS
Hens	Wheat
Hens	Corn
Pigs	Wheat
Pigs	Corn

DATA TABLES:

HAT Harvest/Acres Technology

GRAINS	VALUE
Wheat	1/55
Corn	1/95

HLT Harvest/Labor Technology

GRAINS	VALUE
Wheat	0.22
Corn	0.74

HC Harvest Grains Cost

GRAINS	VALUE
Wheat	0.20
Corn	0.12

BC Buy Grains Cost

GRAINS	VALUE
Wheat	2.50
Corn	1.50

SP Sell Grains Price

GRAINS	VALUE
Wheat	1.75
Corn	0.95

RLT Raise/Labor Technology

ANIMALS	VALUE
Hens	40
Pigs	25

RFT Raise/Floor Technology

GRAINS	VALUE
Hens	15
Pigs	25

RP Raise/Animals Profits

GRAINS	VALUE
Hens	40
Pigs	40

FAT Feed/Animals Technology

ANIMALS	GRAINS	VALUE
Hens	Wheat	0.10
Hens	Corn	0.04
Pigs	Wheat	0.04
Pigs	Corn	0.05

FR Feed/Ratios

ANIMALS	GRAINS	VALUE
Hens	Wheat	-0.20
Hens	Corn	1
Pigs	Wheat	-0.30
Pigs	Corn	1

HLLT Hire-Labor/Labor Technology

VALUE
0.85

HLC Hire-Labor Cost

VALUE
1.50

Acres Supply

VALUE
120

HLS Hire-Labor Supply

VALUE
13,333

LS Labor Supply

VALUE
4,000

FS Floor Supply

VALUE
10,000

Figure 13
RELATIONAL DATA BASE FOR EXTENDED FARMERS PROBLEM

GRAINS:

GRAINS	HAT	HLT	HC	BC	SP
Wheat	1/55	0.22	0.20	2.50	1.75
Corn	1/95	0.74	0.12	1.50	0.95

ANIMALS:

ANIMALS	RLT	RFT	RP
Hens	40	15	40
Pigs	25	25	40

ANIMALS-GRAINS:

ANIMAL	GRAINS	FAT
Hens	Wheat	0.10
Hens	Corn	0.04
Pigs	Wheat	0.04
Pigs	Corn	0.05

HLLT Hire-Labor/Labor Technology

VALUE
0.85

HLC Hire-Labor Cost

VALUE
1.50

AS Acres Supply

VALUE
120

HLS Hire-Labor Supply

VALUE
13,333

LS Labor Supply

VALUE
4,000

FS Floor Supply

VALUE
10,000

Note: Figure excludes values for computed functions, variables, etc.

Figure 14
ELEMENTAL DETAIL TABLES FOR FARMER'S PROBLEM

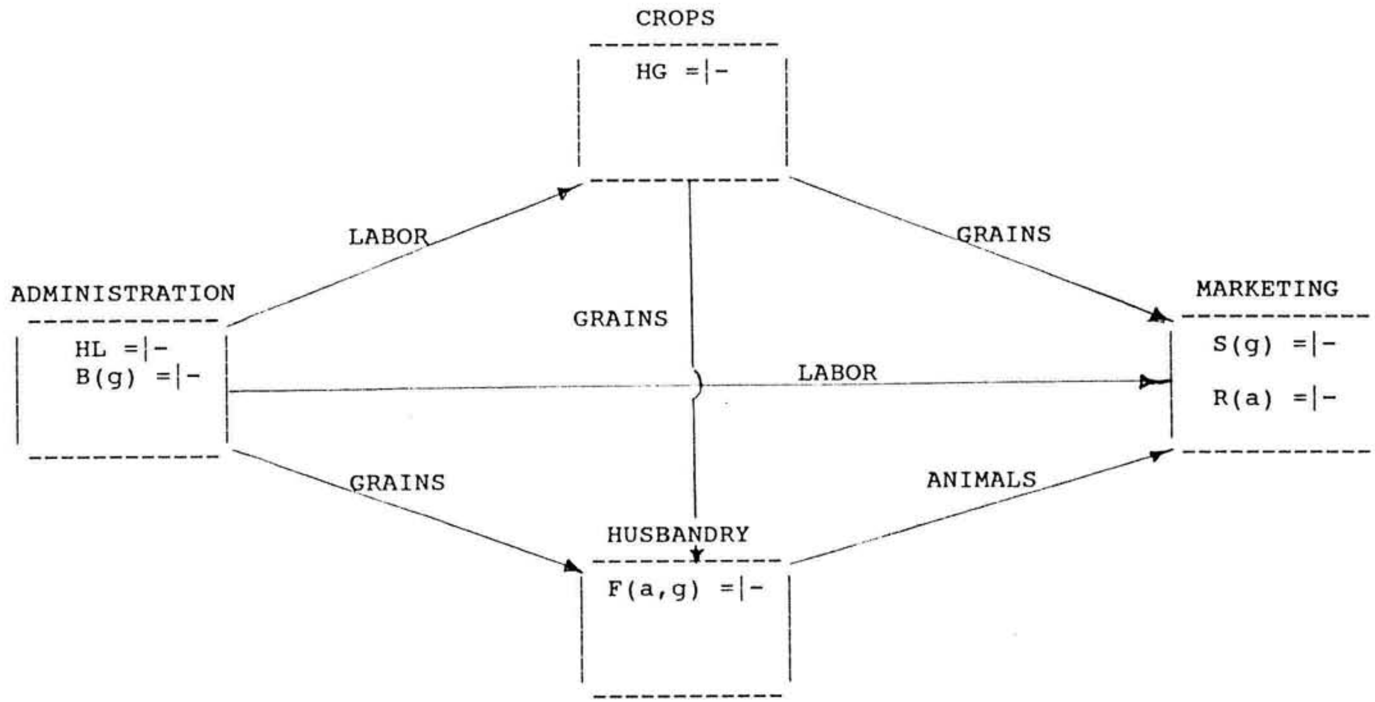


Figure 15
ICONIC REPRESENTATION OF FARMER'S PROBLEM

```
        HIRE-LABOR                          BUY(GRAINS)                        SELL(GRAINS)
        ----------                          ----------                         -----------
            HL                                 B(g)                               S(g)

              |                                  |                                  |
  HIRE-       |                                  |                                  |
  LABOR: 1    |  LABOR: HLLT     DOLLARS: BC     |  GRAINS: 1       GRAINS: 1       |  DOLLARS: SP
------------->|------------->   ------------->|------------->   ------------->|------------->
              |                                  |                                  |
DOLLARS: HLC  |                                  |                                  |
------------->|                                  |                                  |
              |                                  |                                  |

    BOUNDS: [0,HLS]


      HARVEST(GRAINS)                    FEED{ANIMALS,GRAINS}                  RAISE(ANIMALS)
      --------------                     --------------------                  -----------
           H(g)                                 F(a,g)                             R(a)

 ACRES: HAT   |                                  |                                  |
------------->|                                  |                   FLOOR: RFT     |
              |  GRAINS: 1         GRAINS: 1     |  ANIMALS: FAT    ------------->|  DOLLARS: RP
 LABOR: HLT   |------------->     ------------->|------------->                    |------------->
------------->|                                  |                   LABOR: RLT     |
              |                                  |                  ------------->|
DOLLARS: HLC  |                                  |                                  |
------------->|                                  |                   ANIMALS: 1     |
              |                                                     ------------->|

  GEN. BOUNDS: [HLB,HUB]                 RATIOS{ANIMALS}: FR
```

SETS:

NAME
GRAINS
ANIMALS

LIMITS:

COEFFT	CONSTRAINT
AS	Acres Supply
HLS	Hire-Labor Supply
LS	Labor Supply
FS	Floor Supply

Figure 16
LPGRAPH SCREENS FOR FARM PROBLEM DEFINITION

```
LEGEND:
-------

=|-   ACTIVITY

===> FLOW ARC

---> LOGICAL ARC

 .
...   INVENTORY

 _
. .   RESOURCE
 -

=||= MODEL


      WAREHOUSES                    FACTORIES                    MARKETS
     ------------                 ------------                 ------------
    |  B   =|-   |               | PM    =||= |               |  S   =|-   |
    |       .    |   RAW-MATS    |        .   |   PRODUCTS    |            |
    | RM   ...   |==============>| RM    ...  |==============>|            |
    |            |               |        .   |               |            |
    |            |               | PR    ...  |               |            |
     ------------                 ------------                 ------------
```

Figure 17
ICONIC REPRESENTATION OF LP PROBLEMS