# Composing a Structured Model from Validated Submodels for Effective Model Management

Chang-Kyo SUH and Eui-Ho SUH

Management Information Systems Lab
Department of Industrial Engineering, POSTECH
Pohang, 790-600, Korea

Abstract

Structured modeling provides a formal mathematical framework, language, and computer-based environment for conceiving, representing, and manipulating a wide variety of models. It provides a natural framework for integrated modeling owing to its explicit representation power for computational dependencies among submodels. Nevertheless, it doesn't seem to offer a systematic way of composing a structured model from submodels. In order to develop a systematic way, this paper discusses three key issues: (1) Genus structure of validated submodel, (2) Storing method of genus structure, and (3) Integration of genus structures to generate a new genus structure. To visualize the approach, a programming module is developed to implement the step-by-step integration.

*Keywords*: Model Management System, Structured Modeling, Model Integration

## 1. Introduction

The major difference between a Decision Support System (DSS) and other information systems - Transaction Processing System (TPS), Information Reporting System (IRS), and Expert System (ES) - is that a DSS has a model component which provides the user with some alternative choices to make a decision [14]. Thus, successful implementation of a DSS depends upon successful design of modelbase and modelbase management system (MBMS).

The Model Management System (MMS), more generalized term of MBMS, is one of three components in a generalized DSS architecture [12] that performs the tasks of organizing, classifying, accessing and retrieving models from a modelbase in a manner similar to a database management system (DBMS). Various approaches were proposed to formalize the MMS. Some of these approaches include Structured

Modeling approach [5,9,10,11,15], Relational Model approach [1], and Artificial Intelligence (AI) approach [2,4,5,6,7].

Since structured modeling is able to capture the problem semantics of both data models and MS/OR models, it is a natural choice as the modeling framework for the integrated information system architecture for database, DSS, and Management Science (MS) modeling to bridge the gap between the fields of MIS and MS [8]. Geoffrion, in his breakthrough paper [9], describes how one can develop structured modeling. Dolk, however, points out the fact that although structured modeling provides an axiomatic and theoretical basis for model representation as well as a promising medium for increasing recognition of modeling activities within organizations, it is primarily oriented to individual model. To overcome this shortcoming, Dolk [5] extends the Information Resource Dictionary System (IRDS) to accommodate the representation of structured models.

This paper proposes a new approach to overcome the individual orientation of structured models by exploring Geoffrion's fourth guideline for good modeling, that is, composing a model from validated submodels.

This paper is deployed as follows. Section 2 familiarizes the readers with Geoffrion's structured modeling. Section 3 discusses issues on composing a model from validated submodels: (1) Genus structure of validated submodel; (2) Storing method of genus structure; and (3) Integration of genus structures to generate a new genus structure. In Section 4, we summarize the paper and identify a few discussion points.

## 2. Structured Modeling

Structured modeling is a unified modeling framework based on acyclic, attributed graphs to represent cross-references between elements of a model and hierarchies to represent levels of abstraction [9].

Structured modeling frame has three levels: elemental structure, generic structure, and modular structure. Elemental structure aims to capture all of the definitional details of a specific model instance. The generic structure aims to capture the natural familial groupings of elements. The modular structure aims to organize generic structure hierarchically to the extent that this seems appropriate and useful.

With the acyclicity assumption on elemental structure and the monotonicity assumption on modular structure, a structured model can be defined as 1) an elemental structure together with 2) a generic structure satisfying similarity and 3) a monotone modular structure.

As a nongraphical notation for structured modeling, a model schema and elemental detail tables are used. A model schema is formed by expanding syntax (braces denote optionality) [5]:

```
GNAME {index} {calling sequence} /type/ {index set statement}
      {:range statement} {;generic rule} {interpretation}
```

The purpose of elemental detail tables is to describe a particular instance of the generic class of models represented by a schema.

We here provide only minimal material about the objectives and basics of structured modeling. The reader is highly encouraged to consult the references [9,10] for a full, comprehensive details with various examples.

## 3. Composition of Structured Model from Validated Submodels

Lenard [11] and Dolk [5] try to store a single structured model in the relational database framework, but they little discuss the model integration. Geoffrion [9] himself emphasizes the need of model integration. In his words, integrated modeling enables results and insights that cannot be achieved by separate models. Geoffrion, however, addresses the model integration at informal level. A systematic way of integrating multiple models is yet to be provided.

### *3.1 Genus Structure for Model Composition*

Structured modeling process for a single model is summarized as follows:

Step 1: Develop the elemental structure,
Step 2: Develop the genus structure based on an elemental structure, and
Step 3: Develop the modular structure based on a genus structure.

The abstraction level of three structures is hierarchical. First, an elemental structure is a nonempty, finite, closed, acyclic collection of elements. Second, a generic structure is defined on an elemental structure as a collection of partitions, one for each of the five types of elements. Last, a modular structure is defined on a generic structure as a rooted tree whose terminal nodes are in one-to-one correspondence with the genera.

The generic structure is the best candidate to represent the overall structure of model to the model user. The elemental structure has over-detailed information and easily explodes into complexity as the instance of the elements in the model increases, whereas the modular structure is too flexible.

Because there is no mathematical connection between the arcs of the genus graph and those of the modular tree - as a rule of thumb, Geoffrion defines the default modular structure which corresponds to the simplest possible rooted tree with only one module (the root), - various modular structures are available from the unique genus structure. What is more, genus graphs are more attractive devices for managerial communication with the appropriate meaning abstraction and details than the other two structures. Consequently, we propose to store genus structure in order to overcome the individual orientation by storing, updating, and integrating more than one valid submodel into another model.

Based on the genus graphs, structured modeling process can be expanded to compose a structured model from multiple submodels as follows:

Step 1: Develop an elemental, genus, and modular structure for each model,

Step 2: Integrate these genus structures into a single genus structure, and

Step 3: Develop the modular structure based on the integrated single genus structure.

## *3.2 Storage of Structured Models*

In order to work effectively, the MMS must have a rigorous model storage framework. To store the genus structure of each model, we propose a genus relation which has two attributes, that is, called node and calling node. In the relational database framework, the model name, the called node, and the calling node become the table name, the first attribute, and the second attribute, respectively. These two attributes form the composite key.

Definition 1: *Genus relation*

Genus relation (GR) is a binary relation with the called node (Cd) and the calling node (Cg) in the genus graph: GR(Cd, Cg). Each combination of one arc and two nodes connected by that arc is represented by a single tuple in a genus relation.

ModelName(CalledNode, CallingNode)
$Cd \in$ set of elements except root nodes
$Cg \in$ set of elements except primitive entities

The ordering of rows in an array which represents an n-ary relation R is immaterial in the relational data model [3]. We restrict the original property in order to ease converting the genus relation into a genus graph. Instead, all tuples in the genus relation must be ordered tuples.

Definition 2: *Ordered genus relation*

Ordered genus relation, GR'(Cd,Cg), is a binary relation whose tuples are ordered by the following rule.
1) The primitive entity precedes other entities.
2) The CalledNode, Cd, except primitive entity, must be found in the CallingNode, Cg, before it can be found in the CalledNode.
(i.e., $i>j$ for all $Cd_i = Cg_j$, $1 \leq i,j \leq$ cardinality of GR')

## *3.3 Integration of Structured Models*

In the course of model integration, there may be some required information about the given structured models. Some nodes in one model happen to be equivalent to some nodes in the other model, and there are additional nodes necessary to connect the given models in a common sense.

We propose the integration of structured model from the viewpoint of the genus graph.

Step 1: Unify the equivalent nodes from the given genus graphs if there exist at least one pair of equivalent nodes.
Step 2: Convert the additional information into binary relations if there exists additional information to connect the given models in a common sense. Name this genus relation '*Temp*'.
Step 3: Apply the union operation to the given ordered genus relations and '*Temp*' genus relation. Name the new genus relation '*Union*'.
Step 4: Delete every tuple whose first attribute is primitive entity if its second attribute is a node that the genus relation can reach indirectly from the first attribute when the genus relation is expanded in step 6.
Step 5: Make '*Union*' the ordered genus relation.
Step 6: Expand the ordered genus relation to draw the genus graph. In the ordered genus relation, the primitive entity precedes other entities, whereas the primitive entity is the terminal node in the genus graph.

There is one tip to unify the equivalent nodes from the given genus graphs if there exist more than one pair of equivalent nodes. When a compound entity in one model is equivalent to a primitive entity in the other model, the primitive entity is unified with the compound entity, and becomes the compound entity in the integrated genus graph.

## *3.4 Examples*

Figure 1.a shows the genus graph for multi-item EOQ. F, D, Q, and H stands for fixed setup cost, demand rate, order quantity, and holding cost rate respectively. FREQ is a setup frequency. SETUP$ is an annual setup cost. CARRY$ is an annual carrying cost. ITEM$ is an annual item cost. TOT$ is the total annual cost. This genus graph can be stored in the modelbase by means of a binary relation which has 14 tuples as follows.

```
MultiEOQ(ITEM$, TOT$)
MultiEOQ(SETUP$, ITEM$)
MultiEOQ(FREQ, ITEM$)
MultiEOQ(CARRY$, ITEM$)
MultiEOQ(F, SETUP$)
MultiEOQ(FREQ, SETUP$)
MultiEOQ(D, FREQ)
MultiEOQ(Q, FREQ)
MultiEOQ(Q, CARRY$)
MultiEOQ(H, CARRY$)
MultiEOQ(ITEM, F)
MultiEOQ(ITEM, D)
MultiEOQ(ITEM, Q)
MultiEOQ(ITEM, H)
```

---

Figure 1 to be inserted

---

In the genus relation, the primitive entity (eg. ITEM) appears only in the first attribute. The above genus relation must be converted into the ordered genus relation as follows.

MultiEOQ(ITEM, D)
MultiEOQ(ITEM, F)
MultiEOQ(ITEM, H)
MultiEOQ(ITEM, Q)
MultiEOQ(D, FREQ)
MultiEOQ(F, SETUP$)
MultiEOQ(H, CARRY$)
MultiEOQ(Q, FREQ)
MultiEOQ(Q, CARRY$)
MultiEOQ(FREQ, SETUP$)
MultiEOQ(FREQ, ITEM$)
MultiEOQ(SETUP$, ITEM$)
MultiEOQ(CARRY$, ITEM$)
MultiEOQ(ITEM$, TOT$)

After we store the genus graph in the form of the ordered genus relation, the genus graph is easily derived from the bottom (the primitive entity) up to the root node.

Example 1 shows the model integration when some attribute entity has its own structured model to solve more detailed and accurate values. Assume that we have the multi-item EOQ genus graph and Holding Cost Rate genus graph among the various models in the modelbase and intend to integrate these models to make the multi-item EOQ model more accurate holding cost rate. Figure 1.b represents the following four tuples of the ordered genus relation.

HoldingCostRate(ITEM, STORAGE)
HoldingCostRate(ITEM, VAL)
HoldingCostRate(STORAGE, H)
HoldingCostRate(VAL, H)

VAL is a unit value. STORAGE is a storage cost rate. ITEM and H are the same as those of the multi-item EOQ.

There is no need of steps 1 and 2. In step 3, we have 18 tuples after union operation. In step 4, we delete the ModiEOQ(ITEM, H) because its first attribute is primitive entity (ITEM) and its second attribute (H) has indirect path with two tuples, that is, ModiEOQ(ITEM, VAL) and ModiEOQ(VAL, H). Why do we have to delete the tuple ModiEOQ(ITEM, H)? Since the ModiEOQ integrates HoldingCostRate into MultiEOQ and HoldingCostRate explains the more accurate holding cost rate of ModiEOQ, there is no need of ModiEOQ(ITEM, H) any more. After step 5, the ordered genus relation for this integrated genus graph is as follows.

ModiEOQ(ITEM, D)
ModiEOQ(ITEM, F)
ModiEOQ(ITEM, Q)
ModiEOQ(ITEM, STORAGE)
ModiEOQ(ITEM, VAL)
ModiEOQ(D, FREQ)
ModiEOQ(F, SETUP$)
ModiEOQ(Q, FREQ)
ModiEOQ(Q, CARRY$)
ModiEOQ(STORAGE, H)
ModiEOQ(VAL, H)

```
    ModiEOQ(FREQ, SETUP$)
    ModiEOQ(FREQ, ITEM$)
    ModiEOQ(SETUP$, ITEM$)
    ModiEOQ(CARRY$, ITEM$)
    ModiEOQ(H, CARRY$)
    ModiEOQ(ITEM$, TOT$)
```

After step 6, we have Figure 1.c, that is, the complete genus graph which represents the integrated submodels.

Example 2 shows the more complicate case. It explains how to integrate multi-item EOQ model and Transportation problem. Geoffrion [9] explains the need to integrate these two models. "If the transportation model is posed on an annualized basis, solving the usual linear programming problem yields the 'optimal' annual flows, but does not prescribe how often shipments should be made or, equivalently, what the shipment size should be. Very frequent shipments are small and thus good for the customer in that they lead to small inventories, but bad in that they are expensive to receive. Infrequent shipments lead to just the opposite result. The best compromise can be found by solving an EOQ problem for each shipment link... each transportation link plays the role of an item... each transportation flow plays the role of a demand rate."

Figure 2 shows the genus graph for transportation problem. It has 12 tuples in the ordered genus relation.

---

Figure 2 to be inserted

---

According to the tip mentioned in Section 3.3, we unify the MultiEOQ.ITEM with Trans.LINK and MultiEOQ.D with Trans.FLOW respectively in step 1, and rename the MultiEOQ.SETUP$ to REC$ (a shipment receiving cost). In step 2, we have two binary relations: Temp($, TOTCOST) and Temp(TOT$, TOTCOST). After step 5, we have 27 tuples in the ordered genus relation. Figure 3 appears after step 6.

---

Figure 3 to be inserted

---

## 4. Summary

The research described in this paper is performed to provide a systematic way of composing a structured model from validated submodels. An ordered genus relation form is proposed for transforming genus structure. Genus relation and ordered genus relation are formally defined to suggest a relational database style storing method. Union operation is applied to integrate the ordered genus relations to generate an ordered genus relation for a new structured model. We have suggested six steps to integrate the validated submodels using some examples. ComGen - a prototype program to compose genus graphs - is also implemented on an IBM-PC AT compatible machine to validate our approach [13].

Our research only discusses the model composition, however, in some cases, the model decomposition may be necessary. After the submodels have been integrated, selecting an appropriate submodel by decomposition is sometimes very important, but, not that simple. Besides called node and calling node in each genus relation, additional attributes may be required such as a trigger which tells the calling sequence of a submodel. To some extent, model decomposition is as important as model integration.

The implementation we have tried is in a small scale and is ready for a full-scale integration of genus graphs. For the implementation of a full-scale integration, the prototype ComGen needs to be extended. The knowledge base about model integration along with the dictionary for nodes in genus relation will reduce the user's input to initiate the model integration. Also the graphics are recommended. ComGen provides the user with ordered genus relation in a text form. The graphics, however, are better than the text to demonstrate the structure of genus graph.

## REFERENCES

[1] Blanning, R.W., "Issues in the Design of Relational Model Management Systems", *Proceedings of the National Computer Conference*, June 1983, 395-401.

[2] Bonczek, R.H., Holsapple, C.W., and Whinston, A.B., "A Generalized Decision Support System Using Predicate Calculus and Network Data Base Management", *Operations Research*, Vol.29, No.2, March-April 1981, 263-281.

[3] Codd, E.F., "A Relational Model of Data for Large Shared Data Banks", *Communications of the ACM*, Vol.13, No.6, June 1970, 377-387.

[4] Dolk, D.R. and Konsynski, B.R., "Knowledge Representation for Model Management Systems", *IEEE Transactions of Software*, SE-10,6, November 1984, 619-627.

[5] Dolk, D.R., "Model Management and Structured Modeling: The Role of an Information Resource Dictionary System", *Communication of the ACM*, Vol.31, No.6, June 1988, 704-718.

[6] Dutta, A. and Basu, A.,"An Artificial Intelligence Approach to Model Management in Decision Support Systems", *Computer*, Vol.17, No.9, September 1984, 89-97.

[7] Elam, J.J. and Henderson, J.C., "Knowledge Engineering Concepts for Decision Support System Design and Implementation", *Information & Management*, Vol.6, 1983, 109-114.

[8] Farn, C.K., "An Integrated Information System Architecture Based on Structured Modeling", *Unpublished Ph.D. Dissertation*, Graduate School of Management, UCLA, 1985.

[9] Geoffrion, A.M., "An Introduction to Structured Modeling", *Management Science*, Vol.33, No.5, 1987, 547-588.

[10] Geoffrion, A.M., "The Formal Aspects of Structured Modeling", *Operations Research*, Vol.37, No.1, Jan.-Feb. 1989, 30-51.

[11] Lenard, M.L., "Representing Models as Data", *Journal of Management Information Systems*, Vol.2, No.4, 1986, 36-48.
[12] Sprague, R.H. and Carlson, E.D., *Building Effective Decision Support Systems*, Prentice-Hall, 1982.
[13] Suh, Chang-Kyo and Suh, Eui-Ho., "Composing a Structured Model from Validated Submodels for Effective Model Management", *Technical Report IE-TR-90-02*, POSTECH, January 1990.
[14] Suh, Eui-Ho and Hinomoto, H., "Use of A Dialogbase For Integrated 'Relational' Decision Support Systems", *Decision Support Systems*, Vol.5, No.3, 1989, 277-286.
[15] Suh, Eui-Ho, Suh, Chang-Kyo, and Brian LeClaire., "Object-Oriented (O-O) Structured Modeling for an O-O DSS", *Technical Report IE-TR-89-08*, POSTECH, November 1989.

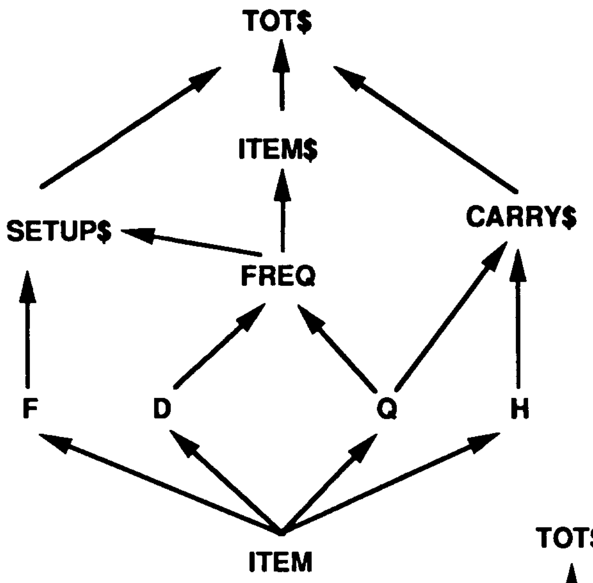


Figure 1.a Genus Graph for multi-Item EOQ

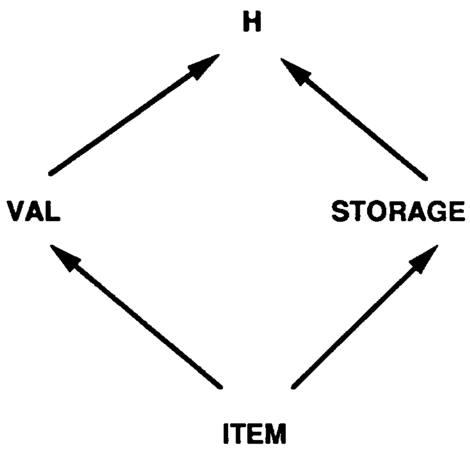


Figure 1.b Genus Graph for Holding Cost Rate

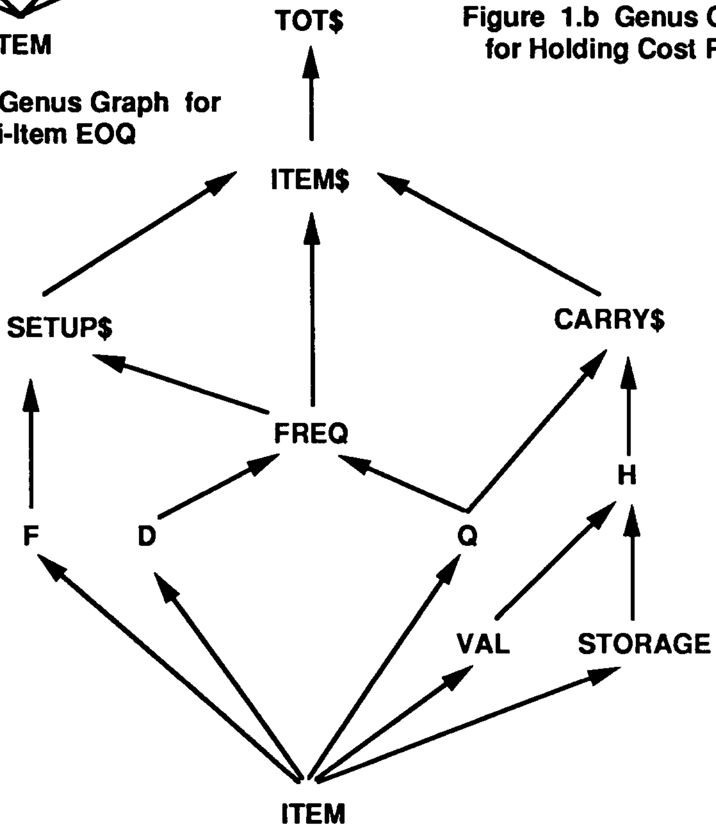
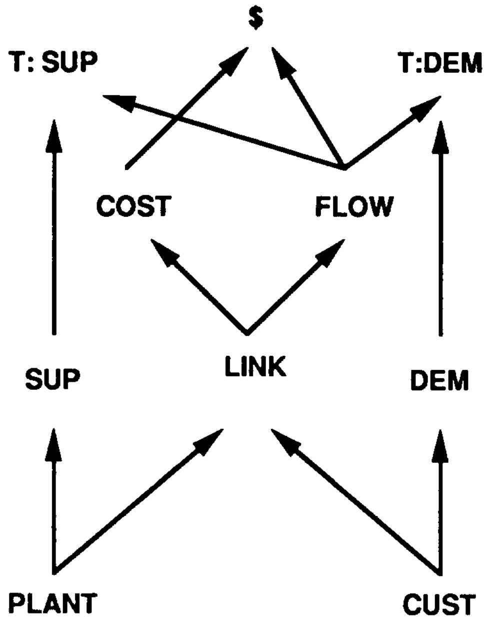


Figure 1.c Genus Graph for Modified EOQ

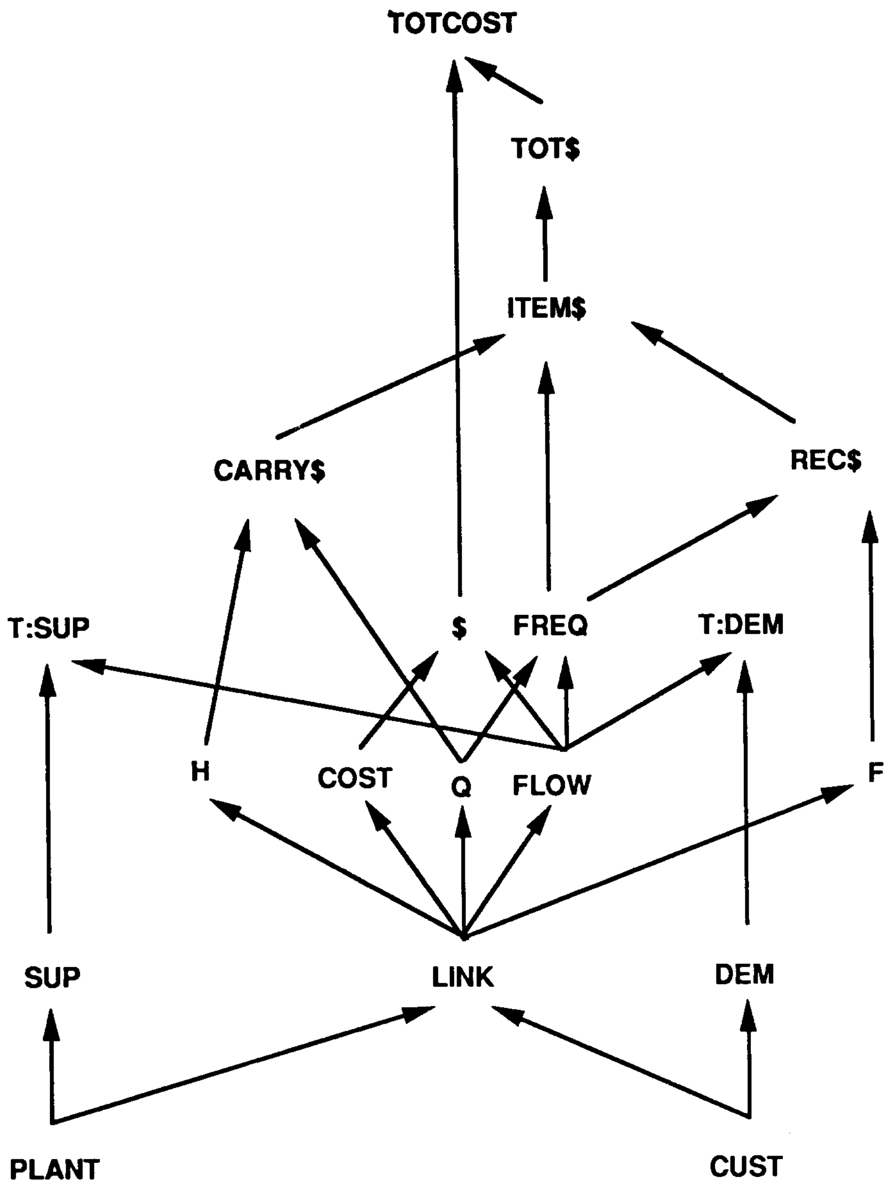
Figure 1 Genus Graphs for Example 1



**Transportation**

(PLANT, SUP)
(PLANT, LINK)
(CUST, LINK)
(CUST, DEM)
(SUP, T:SUP)
(LINK, COST)
(LINK, FLOW)
(DEM, T:DEM)
(COST, $)
(FLOW, T:SUP)
(FLOW, $)
(FLOW, T:DEM)

Figure 2 Genus Graph and Ordered Genus Relation for Transportation



**Integrated**

(PLANT, SUP)
(PLANT, LINK)
(CUST, LINK)
(CUST, DEM)
(SUP, T:SUP)
(LINK, H)
(LINK, COST)
(LINK, Q)
(LINK, FLOW)
(LINK, F)
(DEM, T:DEM)
(H, CARRY$)
(COST, $)
(Q, CARRY$)
(Q, FREQ)
(FLOW, T:SUP)
(FLOW, $)
(FLOW, FREQ)
(FLOW, T:DEM)
(F, REC$)
(CARRY$, ITEM$)
($, TOTCOST)
(FREQ, ITEM$)
(FREQ, REC$)
(REC$, ITEM$)
(ITEM$, TOT$)
(TOT$, TOTCOST)

Figure 3 Genus Graph and Ordered Genus Relations for Integrated Model