Proceedings of the 1987 Winter Simulation Conference
A. Thesen, H. Grant, W. David Kelton (eds.)

CONSTRAINT-DRIVEN GENERATION OF MODEL STRUCTURES

Jerzy W. Rozenblit and Yueh-Min Huang

Dept. of Electrical and Computer Engineering
University of Arizona
Tucson, Arizona 85721

ABSTRACT

This article presents a framework for generating model structures with respect to a set of constraints and modelling requirements. The framework is based on multifacetted modelling and artificial intelligence concepts. Two knowledge representations, the system entity structure and the production rule formalism are incorporated into an automatic procedure for generating model configurations. The procedure is implemented in the Turbo Prolog environment. A simple case study based on a local area network (LAN) modelling problem is discussed to illustrate the conceptual framework.

BACKGROUND

The concepts of model development presented here are derived from multifacetted modelling methodology (Zeigler, 1984). Multifacetted methodology denotes a modelling approach which recognizes the existence of multiplicities of objectives and models in any simulation project. It provides formal representation schemes that support the modeller in organizing the model construction process, aggregating partial models, and in specifying simulation experiments (Zeigler, 1984).

The key concept underlying structuring of models, their organization, and specification of simulation experiments (experimental frames) is the system entity structure (Zeigler, 1984). The system entity structure is based on a tree-like graph that encompasses the boundaries, decompositions and taxonomic relationships that have been perceived for the system being modelled. An entity signifies a conceptual part of the system which has been identified as a component in one or more decompositions. Each such decomposition is called an aspect. Thus entities and aspects are thought of as components and decompositions, respectively. In addition to decompositions, there are relations termed specializations. A specialization relation facilitates representation of variants for an entity. Called specialized entities, such variants inherit properties of an entity to which they are related by the specialization relation.

Entities have attributes represented by the attached variable types. When a variable type V is attached to an entity E, this signifies that a variable I.E may be used to describe a property of the entity E.

Aspects can have coupling constraints attached to them. Coupling constraints restrict the way in which components (represented by entities) identified in decompositions (represented by aspects) can be joined together.

In addition to coupling constraints, there are selection constraints in the system entity structure. Selection constraints are associated with specializations of an entity. They restrict the way in which its subentities may replace it in the process of model construction (Rozenblit et. al., 1986).

The other fundamental concept underlying the multifacetted framework is the experimental frame (Zeigler, 1984). Briefly, an experimental frame defines a set of input, control, output, and summary variables, and input and control trajectories. These objects specify conditions under which a model can be observed and experimented with.

The experimental frame concept has been generalized by Rozenblit and Zeigler (1985, 1986, 1987) who introduced the generic experimental frame definition. A generic experimental frame consists of input, output, and summary generic variable types. The variable types express performance indices associated with a given modelling objective. The modeller should proceed as follows in order to define a generic frame: first he should identify modelling objectives. With each objective he should specify performance indices that will provide measures of the objective realization by the simulation model. In the next phase, a set of generic variable types that will allow the modeller to obtain the performance indices should be defined. These variables specify a generic frame. The reader is referred to (Rozenblit and Zeigler, 1985, 1987; Rozenblit et.al., 1986) for examples of generic frame definitions.

Given the system entity structure the modeller has a choice of a number of model alternatives. This is due to the multiplicity of aspects and specializations. Thus, we require that the modeller have procedures for generating model structures pertaining to the modelling objectives. Such structures should be selected from the system entity structure.

In our previous research we have developed algorithms that prune the system entity structure with respect to a generic experimental frame (Rozenblit, 1986). Generic frames represent behavioral (performance) aspects of the modelling objectives. Therefore it is natural to seek substructures of the system entity structure that possess attributes expressed in a generic experimental frame. If such substructures are found then we can say that models constructed from them realize the modelling objective expressed by the generic frame.

The search process for the substructures that realize a generic frame proceeds as follows: every entity in each aspect of the system entity structure is searched for occurrences of variable types present in the generic frame. The entities whose attached variable types match those in the generic frame are used to build the model composition tree (Zeigler, 1984). The model composition tree is a basis for hierarchical model development. Whenever there is a specialization,

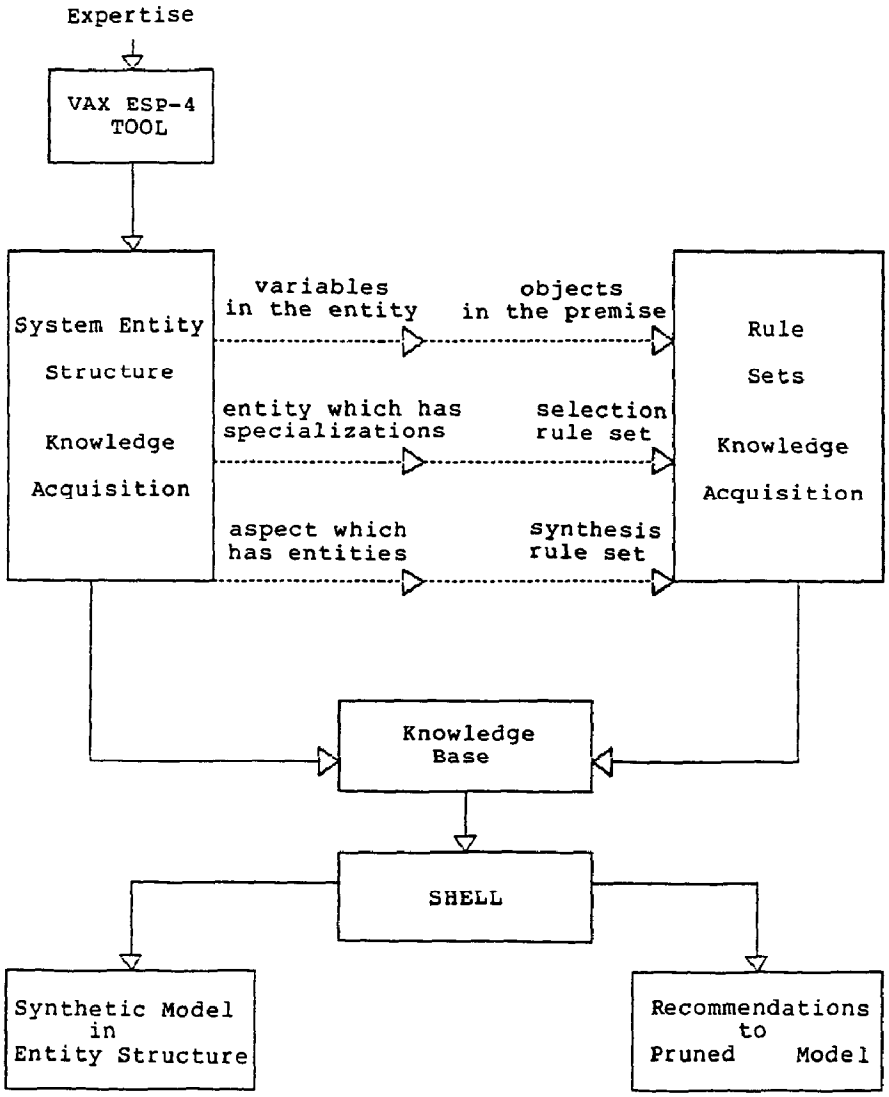

Figure 1. Model Structure Generator System

a choice of a unique entity must be made. In this paper we focus on the process that employs the production rule formalism to support automatic selection of entities from taxonomic relationships and synthesis of structures underlying the simulation models. We term this process constraint-driven system entity structure pruning.

The process consists in specifying the system entity structure for a given modelling problem. Then, a knowledge base that contains rules for selection and configuration of the entities is constructed. The rules are derived from both the requirements of the project and its constraints. Also, the knowledge acquisition process soliciting relevant information from experts in a given problem domain is employed to define the rules.

The modeller invokes the inference engine which, through a series of queries based on the constraint rules, allows him/her to consult on an appropriate structure for the modelling problem at hand.

RULE-BASED MODEL STRUCTURE SYNTHESIS

We now proceed to describe the system we have developed to automatically generate model structures. The system whose architecture is depicted in Figure 1 consists of a knowledge base and an inferencing shell that generates recommendations for model structure synthesis.

Knowledge Base Construction

The process of knowledge base construction begins with setting up the system entity structure for the model being constructed. At the present time we use previously developed tools for entity structuring (ESP4 - Entity Structuring Program (Zeigler et. al., 1980)). The system entity structure is a basis for what we term a conceptual network. This is a declarative representation of modelling domain objects.

From the standpoint of problem-solving processes,

model structure generation can be interpreted as a search through a space of solution states. New "model states" evolve through a process of analysis, synthesis, evaluation, and regeneration. The production rule formalism serves as a basis for our model generation framework.

There are several advantages to using the production rule scheme: a.) the conversion of knowledge into a rigid formalism results in easy checking of uniformity; b.) each production rule represents a small, independent piece of knowledge - this facilitates modularity; c.) rigid syntax affords the convenience of checking consistency; d.) it is easy to furnish explanation facilities (Winston, 1984; Nilsson, 1980).

In our system, the production rule formalism is used to express modelling objectives and constraints. In the detailed LAN example, we shall show how the constraints are represented in rule sets.

To prune the system entity structure, we generate the following rule sets:

Selection rule set: each selection rule stands for a choice of an entity in a specialization.

Synthesis rule set: after selection rules have been applied to the entity structure, synthesis rules ensure proper configuration of the selected entities. They also coordinate the actions of the selection rules. Certainty factors are are employed to indicate the applicability of the rules.

The constraint rule base is built according the guidelines given below:

Phase 1: Selection Rules

a.) attached variables of an entity are treated as objects. They are included in the premise parts of the rules. Their legal values are indicated according to the expertise acquired from the modeller (or another expert).

b.) conclusion parts of the rules contain the specialized entities of the entity from step a. If needed, certainty factors are assigned to each rule.

c.) for another entity in the same aspect step a is repeated.

d.) steps a, b, c, are repeated until every aspect has been assigned rules.

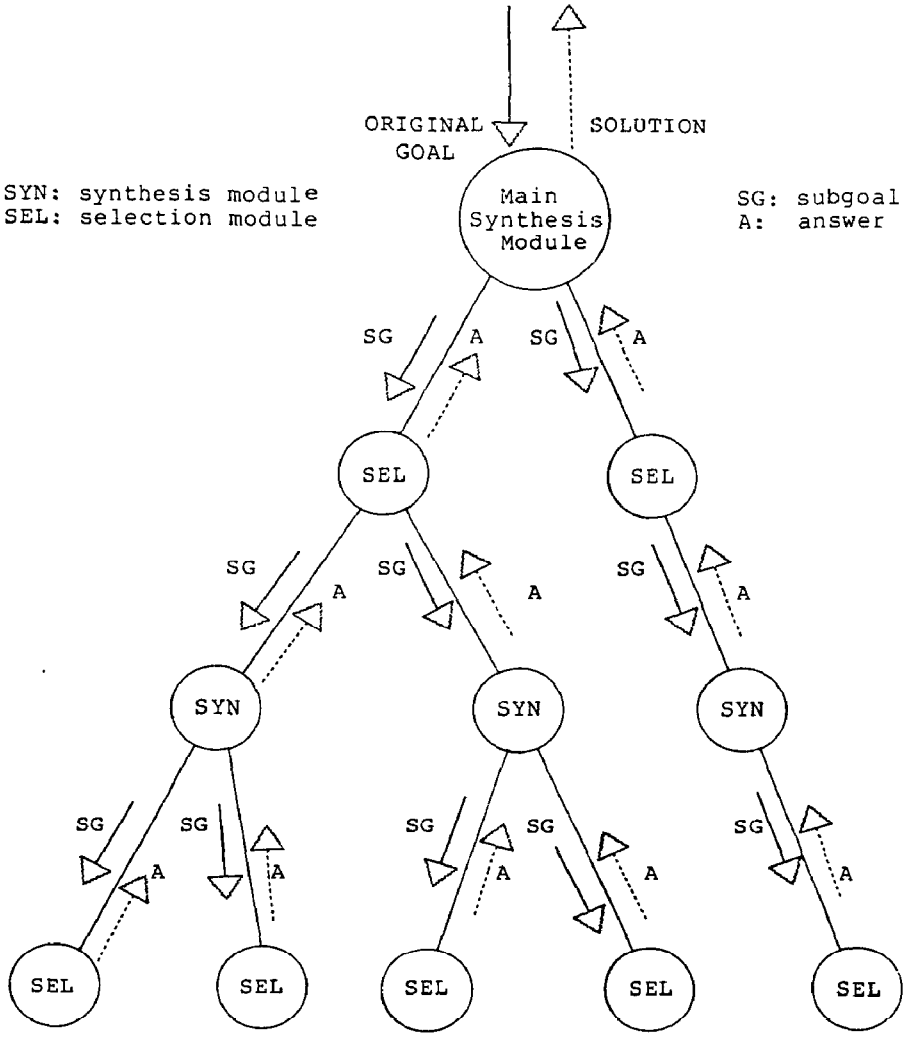

Figure 2. System's Inferencing Scheme

Phase 2: Synthesis Rules

a.) compatible objects from conclusion parts of the selection rules are included in the premise parts

b.) recommendations are given for the above objects in the conclusion parts of the rule.

Selection rules are associated with the entities whereas the synthesis rules are attached to the aspects of the domain entity structure. Each rule set can be regarded as a module. Therefore the entire rule base is constructed in a hierarchical manner imposed by the entity structure. We believe such a hierarchical structure is necessary to increase the efficiency of pruning in systems with a large number of rules.

Rule Syntax

In order to reduce the number of links between modules in the hierarchically organized rule base, we allow for multiple actions (conclusions) in the rule syntax. To reduce the number of modules, we connect the premises with the logical "or" or "and". The template rule syntax has the following form:

```
if object_attribute_1 = value_1 and/or
   object_attribute_2 = value_2 and/or
                ...
                ...
   object_attribute_n = value_n

then conclusion_1 = value_1 (cf1) and
     conclusion_2 = value_2 (cf2) and

                ...
                ...
```

where cf1,cf2,..., are certainty factors whose values range from 0 which stands for no recommendation, to 1 which denotes a strong recommendation.

Inference Engine Design

The system's shell has been implemented in Turbo Prolog and runs on IBM PC compatible machines. The inference engine uses the strategy of "generate and test", i.e., it takes the initial data from the user and the hypothesis generated by the knowledge base to prune the search space tree. In other words, the engine attempts to match the data with the information contained in the knowledge base. If the data match, the engine climbs up the tree, trying to prove the next hypothesis, as shown in Figure 2. We use aspect ordering in order to eliminate aspects not desirable in the model we are constructing, and specialization-oriented pruning to select unique entities for the model composition trees.

For details, concerning the inferencing mechanism we refer the reader to Huang (1987).

User Interface

We use multiple windows in the user interface. There are two basic windows: entity structure display and consultation display. The former is in the form of a tree which can be perused in any manner. The latter is a menu-driven window.

The values of objects' attributes are retrieved from the constraint rule base automatically. Besides the values, other terms such as UNKNOWN, WHAT, WHY are included in the menu. They provide explanation facilities as to what fact has been determined and what is the trace of rules that have been used.

The system is still at the development phase. We are currently improving the user interface and designing a procedure that will automatically link the knowledge base and the shell with the entity structuring tool.

Having provided a brief description of the model structure generating system, we now proceed to illustrate its operation on a simple example from the area of local area network modelling.

EXAMPLE: GENERATING A LAN MODEL STRUCTURE

In our previous work (Rozenblit et.al., 1986, Sevinc, 1986) we have used local area networks to verify our theoretical results concerning knowledge-based modelling and simulation. Here, we present a simple example illustrating the ideas presented in the foregoing sections.

The entity structure shown in Figure 3 presents a family of possible structures for a model of a local area network. The number of choices are given by the specialization relations. For example, one might select a model with a bus topology, optical fiber, or coaxial cable transmission medium.

In our example, the entity LAN has two aspects: functional partition and medium access control, and one topology specialization. Furthermore, entities identified in those aspects are classified into more specialized objects e.g., transmission medium in functional partition specializes into a coaxial cable, optical fiber, and twisted pair. Each object of the LAN conceptual network has attributes. They are preceeded by the symbol "-", as shown in Figure 3.

The structuring of the system is the first step in our framework. The next step is the construction of the knowledge base. We have defined the selection and synthesis rules on the basis of consultations with LAN expert designers and literature studies (Fritz et.al., 1985; Hawe et.al., 1984; Hutchinson et. al., 1985; Madron, 1984; Stallings, 1984; Tannenbaum, 1981). We have assumed legal values for attributes to be high, medium, and low. Certainty factors have been assigned based on the acquired expertise. The following are the rules for the LAN entity structure of Figure 3.

Rules to select LAN topology

```
rule 1
if flexibility = high or reliability = high
then recommend_LAN = bus_of_LAN (0.9)

rule 2
if throughput = high and reliability = medium or
   flexibility = medium
then recommend_LAN = ring_of_LAN (0.9)

rule 3
if throughput = medium and reliability = low or
   flexibility = low
then recommend_LAN = star_of_LAN (0.9)
```

Rules to select protocol

```
rule 4
if access reliability = high and data rate = low or
   packet delay time = high
then recommend access protocol = CSMA/CD (1.0)
```

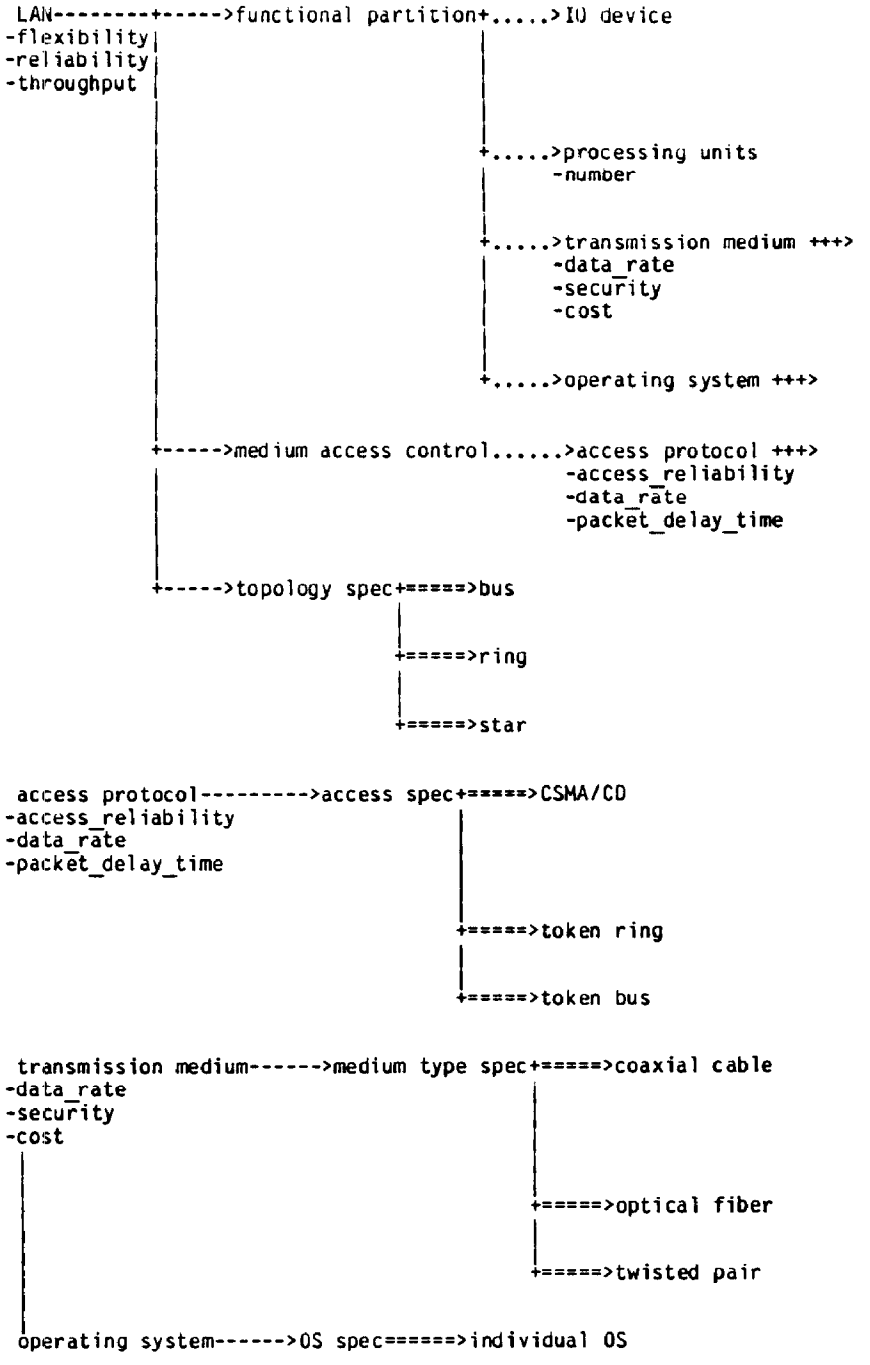

Figure 3. LAN Entity Structure

```
rule 5
if access reliability = medium and data rate = high or
   packet delay time = medium or
   packet delay time = low
then recommend access protocol = token_bus (0.9) and
     recommend access protocol = token_ring (0.8)
```

Rules to select medium

```
rule 6
if data rate = low and security = low or cost = low
then recommend transmission medium = twisted_pair
(1.0)

rule 7
if data rate = medium or data rate = high and
   security = medium and cost = medium
then recommend trans. medium = coaxial cable (1.0)

rule 8
if security = high or cost = high
then recommend trans. medium = optical fiber (1.0)
```

Rules for model structure synthesis

```
rule 9
if recommend LAN = bus_of_LAN and
   recommend access protocol = CSMA/CD and
   recommend trans. medium = twisted pair
then LAN = bus of LAN (0.9) and
     medium access control = CSMA/CD (0.9) and
     I/O devices = no_recommendation (0.9) and
     processing unit = 10_in_number (1.0)  and
     trans. medium = twisted pair (0.9)
     operating system = individual OS (1.0)

rule 10
if recommend LAN = bus_of_LAN and
   recommend access protocol = token bus and
   recommend trans. medium = coaxial cable
then LAN = bus of LAN (0.9) and
     medium access control = token bus  (0.9) and
     I/O devices = no_recommendation (0.9) and
     processing unit = varied 10_to_1000 (1.0)  and
     trans. medium = coaxial cable (0.9)
     operating system = individual OS (1.0)

rule 11
if recommend LAN = ring_of_LAN and
   recommend access protocol = token ring and
   recommend trans. medium = coaxial cable
then LAN = ring LAN (0.9) and
     medium access control = token ring (0.9) and
     I/O devices = no_recommendation (0.9) and
     processing unit = varied 10_to_1000 (1.0)  and
     trans. medium = coaxial cable (0.9)
     operating system = individual OS (1.0)

rule 12
if recommend LAN = ring_of_LAN and
   recommend access protocol = token ring and
   recommend trans. medium = optical fiber
then LAN = ring of LAN (0.9) and
     medium access control = token ring (0.9) and
     I/O devices = no_recommendation (0.9) and
     processing unit = 10_in_number (1.0)  and
     trans. medium = optical fiber (0.9)
     operating system = individual OS (1.0)


rule 13
if recommend LAN = star_of_LAN and
   recommend trans. medium = twisted pair
then LAN = star of LAN (0.9) and
     medium access control =
     no_recommendation (0.0) and
     I/O devices = no_recommendation (0.9) and
     processing unit = 10_in_number (1.0)  and
     trans. medium = twisted pair (0.9) and
     operating system = individual OS (1.0)
```

The above rules express both selection and synthesis (configuration) constraints for constructing our simple LAN model. Having set up the knowledge base, we seek recommendations for generating model structures. This is accomplished through consultation sessions with the system. An example session produced the following recommendation: (with respect to the modelling requirements expressed through the system's queries)

LAN Consultation:

```
What is the extent of flexibility?
HIGH

What is the demand on access reliability?
MEDIUM

What is the data rate demand?
MEDIUM

What is the allowable packet delay time?
LOW

What is the desired transmissive security?
MEDIUM

What is the budget?
MEDIUM
```

Conclusion:

```
Fired Rules <1><5><7><10>

LAN is bus_of_LAN (0.81)
medium access control is token bus (0.81)
I/O device is no_recommendation (0.0)
processing units is varied 10_to_1000 units (0.9)
transmission medium is coaxial cable (0.81)
operating system is individual OS (0.9)
```

The pruned entity structure recommended by the above consultation session is given in Figure 4. By pruning the system entity structure with respect to the constraint knowledge base we ensure satisfaction of the requirements, and at the same time, we automatically restrict the space of alternative model structures that may be used for model construction.

CONCLUSIONS

This paper further extends our research into the methodology of model development. We have augmented system entity structure pruning algorithms with a rule-based process for selecting and synthesizing model objects representing model components. This process is driven by the modelling project's requirements and constraints. Therefore, we are now able to assist the modeller in choosing and properly configuring the model components. Viewed from the validation perspective, our framework provides a means of establishing preliminary validity of the model in terms of its structural conformance to the constraints and requirements.

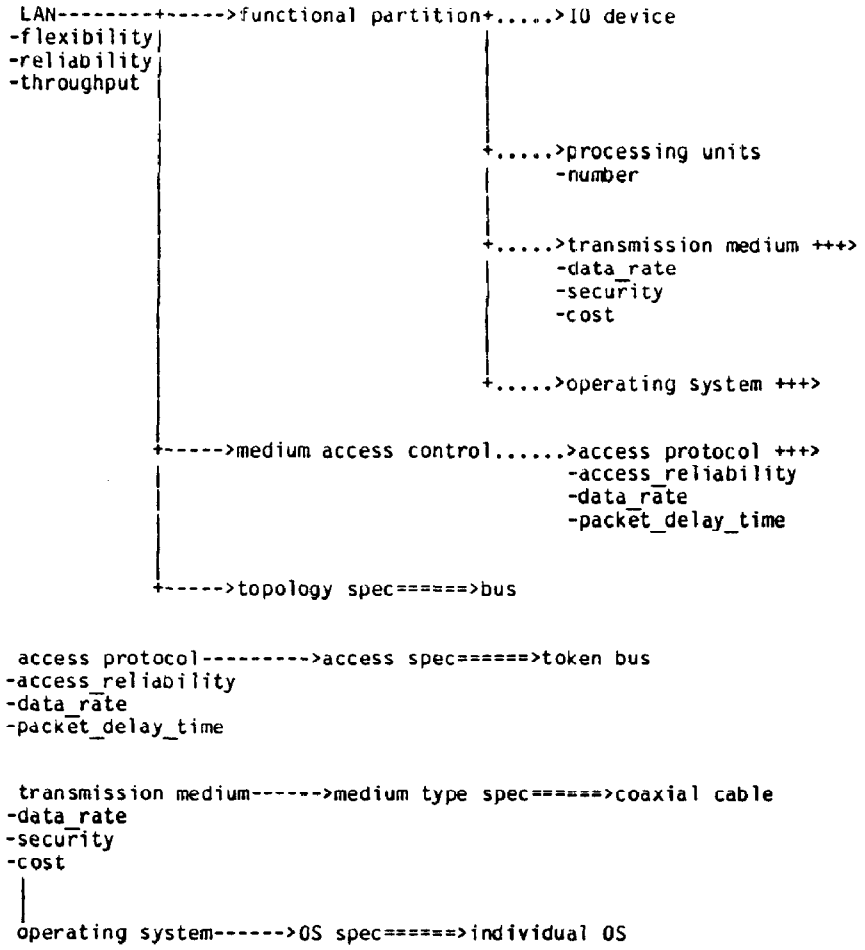

Figure 4. Pruned LAN Entity Structure

Our current efforts are focused on reducing the complexity of the knowledge base resulting from the number of rules that have to be specified for a given entity structure. This complexity can be partly reduced by employing the rule syntax discussed above, and by restricting the size of the system entity structure. The latter can be accomplished by what we term domain pruning in which aspects irrelevant to the modelling objectives are eliminated prior to the rule base specification. We shall report on the developments in this direction in the future.

REFERENCES

Fritz, J.S. et. al. (1985) "Local Area Networks, Selection Guidelines", Prentice-Hall, Engelwood Cliffs, New Jersey.

Hawe, B., et. al. (1984) "Transparent Interconnection of local area networks with bridges", Journal of Telecommunication Networks, 3(2), pp. 116-129.

Huang, Y.M., (1987) Building an Expert System Shell for Model Synthesis in Logic Programming", M.S. thesis, Dept. of Electrical and Computer Engineering, The University of Arizona, Tucson, AZ

Hutchinson D., et. al. (1985) "Local Area Networks, an Advanced Course", Springer-Verlag.

Madron, T.W., (1984) "Local Area Networks in Large Organizations", Hayden Book Co., New Jersey.

Nilsson, N.J., (1980) "Principles of Artificial Intelligence", Tioga, Palo Alto, CA.

Rozenblit, J.W., Zeigler, B.P., (1985) "Concepts for Knowledge-Based System Design Environments", Proc. of the 1985 Winter Simulation Conference, San Francisco, CA.

Rozenblit, J.W., Zeigler, B.P. (1987) "Design and Modelling Concepts", Encyclopedia of Robotics, John Wiley, N.Y.

Rozenblit, J.W., Zeigler B.P., (1986) "Entity-Based Structures for Model and Experimental Frame Construction", in "Modelling and Simulation in Artificial Intelligence Era" (ed. M.S. Elzas et. al.) North Holland, Amsterdam.

Rozenblit, J.W. (1986) "A Conceptual Basis for Integrated, Model-Based System Design", Technical Report, Dept. of Electrical and Computer Engineering, University of Arizona, Tucson, January 1986.

Sevinc, S., (1986) "Entity Structure Representation for Local Area Network Simulation", MS Thesis, Dept. of Electrical and Computer Engineering, University of Arizona, Tucson, AZ.

Stallings, W. (1984) "Local Networks", MacMillan Pub. Co., New York.

Tannenbaum, A. S., (1981) "Computer Networks", Prentice-Hall, Engelwood Cliffs, New Jersey.

Winston, P.H., (1984) "Artificial Intelligence", Addison-Wesley, Reading, MA.

Zeigler, B.P., Belogus D., Bolshoi, A., (1980) "ESP - An Interactive Tool for System Structuring", Proc. of the 1980 European Meeting on Cybernetics and Systems Research, Hemisphere Press.

Zeigler, B.P. (1984) "Multifacetted Modelling and Discrete Event Simulation", Academic Press, London.

JERZY W. ROZENBLIT is an assistant professor in the Electrical and Computer Engineering Department at The University of Arizona. He received his Ph.D. in Computer Science from Wayne State University in Detroit, in 1985. His research interests are in the areas of modelling and simulation, system design, and artificial intelligence. He is a member of ACM, IEEE Computer Society, and The Society for Computer Simulation.

Jerzy W. Rozenblit
Dept. of Electrical and Computer Engineering
The University of Arizona
Tucson, Arizona 85721
(602) 621-6177

YUEH MIN HUANG is a graduate student in the Department of Electrical and Computer Engineering at The University of Arizona. He received his BS degree in Engineering Science from the National Cheng Kung University in Tainan, Taiwan. His areas of research are artificial intelligence, expert systems, and engineering design. He is a member of American Association for Artificial Intelligence.

Yueh Min Huang
Dept. of Electrical and Computer Engineering
The University of Arizona
Tucson, Arizona 85721
(602) 621-2434