THE DESIGN OF AN AUTOMATIC CONSULTANT: AN OVERVIEW

Hassan Ait-Kaci

OR 900
Individual Study
Spring 1980

INTRODUCTION

This paper proposes an overview of a possible model management system design with an emphasis on the subsystem dealing with a user's problem structuring. It goes along the lines of the general ideas discussed by J. Elam [8] as far as making modelling an integral part of the system via a concept processing facility. A set of potential ways for solving issues raised by J. Elam is investigated using a semantic network structure borrowed from R. Brachman [2] after a description of a decision support system functions and components. The purpose here is to prepare for what I believe to be a fruitful and promising opening in model management system design research. Thus, the density of the material presented and the relative lack of illustrative explanations come from the base setting aspect of the work.

The author is solely responsible for the content of this paper reflecting his work for an individual study taken during the spring term 1980, under the supervision of Dr. Joyce Elam at the Decision Sciences Department of the Wharton School. He is indebted to Dr. J. Elam and Dr. J. Henderson who kindly listened to early ideas and encouraged him to pursue the work.

OR/MS CONCEPTUAL MODELLING

In the world of decision making, the methodology of modelling has become central as it represents the modern way of analysis. As a consequence, management science is universally understood as modelling science. What a model is, should be, or should not be has hence constituted a harsh subject of argument, still going on. But, as things in this dynamic world have to somehow move forth, the lack of universal definition for it does not prevent everyone from dealing with it on a daily basis with a lot of comfort. Everyone has his/her own idea of what a "model" is.

Now, models represent some description of something which needs to be understood, analyzed, decided, etc., and do not exist in the nature to be chased and captured; they have to be built.

Modelling is building models. Unfortunately, it can be done in numerous ways, making the science as difficult as it is known to be. However, decision making has the particularity of facing situations and systems that present some types of recurrences and/or resemblances which have enabled scientists and researchers to construct and classify models and representations with well understood methods and techniques. The model builder's job is now to jostle around with a real world situation and find an appropriate method for it; i.e., filling a set of assumptions and satisfying a set of properties and objectives, subject to and working within the model's synergy. This is called consultation: finding a path in an intricate conceptual jungle from a real world to a

complex maze of techniques and methods, through assumptions, definitions, descriptions, deductions and inductions, and back. Tough job, as I already stated it.

Fortunately -- but not too much --, consultants do it by explicitly or implicitly categorizing, ordering, associating, etc., the two ends of their rainbows: the models on one hand, and the usage of these models on the other hand.

Indeed, a taxonomy of models according to their similarities or contrasts or other conceptual relationships, and a taxonomy of circumstances of use of the methods simplify the consultant's task and provides him/her with templates on the grounds of which to decide which is adequate to what.

To illustrate this, let's consider figure 1 giving an example of a taxonomy of methods in the field of multiple criteria decision making, borrowed from [10]. The models are categorized explicitly according to their respective approaches. As for the specifications of uses of these methods, figure 2, drawn from the same reference, illustrates a taxonomy of uses which determines the adequacy for some approaches based on a set of assumptions and features. What figure 2 represents is in fact the skeleton of a consultant's modelling methodology in the field of multiple criteria decision making. It is essentially a query of basic concepts that have definite mapping potentialities in the consultant's mind towards a context of models. This is the characteristic that I would like to emphasize here.

Multiple Objective/Multiple Attribute Decision Methods

A. Weighting Methods
   1. Inferred preferences
      a. Linear regression
      b. Analysis of variance
      c. Quasi-linear regression
   2. Directly assessed preferences: general aggregation
      a. Trade-offs
      b. Simple additive weighting
      c. Hierarchical additive weighting
      d. Quasi-additive weighting
   3. Directly assessed preferences: specialized aggregation
      a. Maximin
      b. Maximax

B. Sequential Elimination Methods
   1. Alternative versus standard: comparison across attributes
      a. Disjunctive and conjunctive constraints
   2. Alternative versus alternative: comparison across attributes
      a. Dominance
   3. Alternative versus alternative: comparison across alternatives
      a. Lexicography
      b. Elimination by aspects

C. Mathematical Programming Methods
   1. Global objective function
      a. Linear programming
   2. Goals in constraints
      a. Goal programming
   3. Local objectives: interactive
      a. Interactive, multi-criterion programming

D. Spatial Proximity Methods
   1. Iso-preference graphs
      a. Indifference map
   2. Ideal points
      a. Multi-dimensional, non-metric scaling
   3. Graphical preferences
      a. Graphical overlays

Figure 1 - A Taxonomy of Models

Method Specification Chart

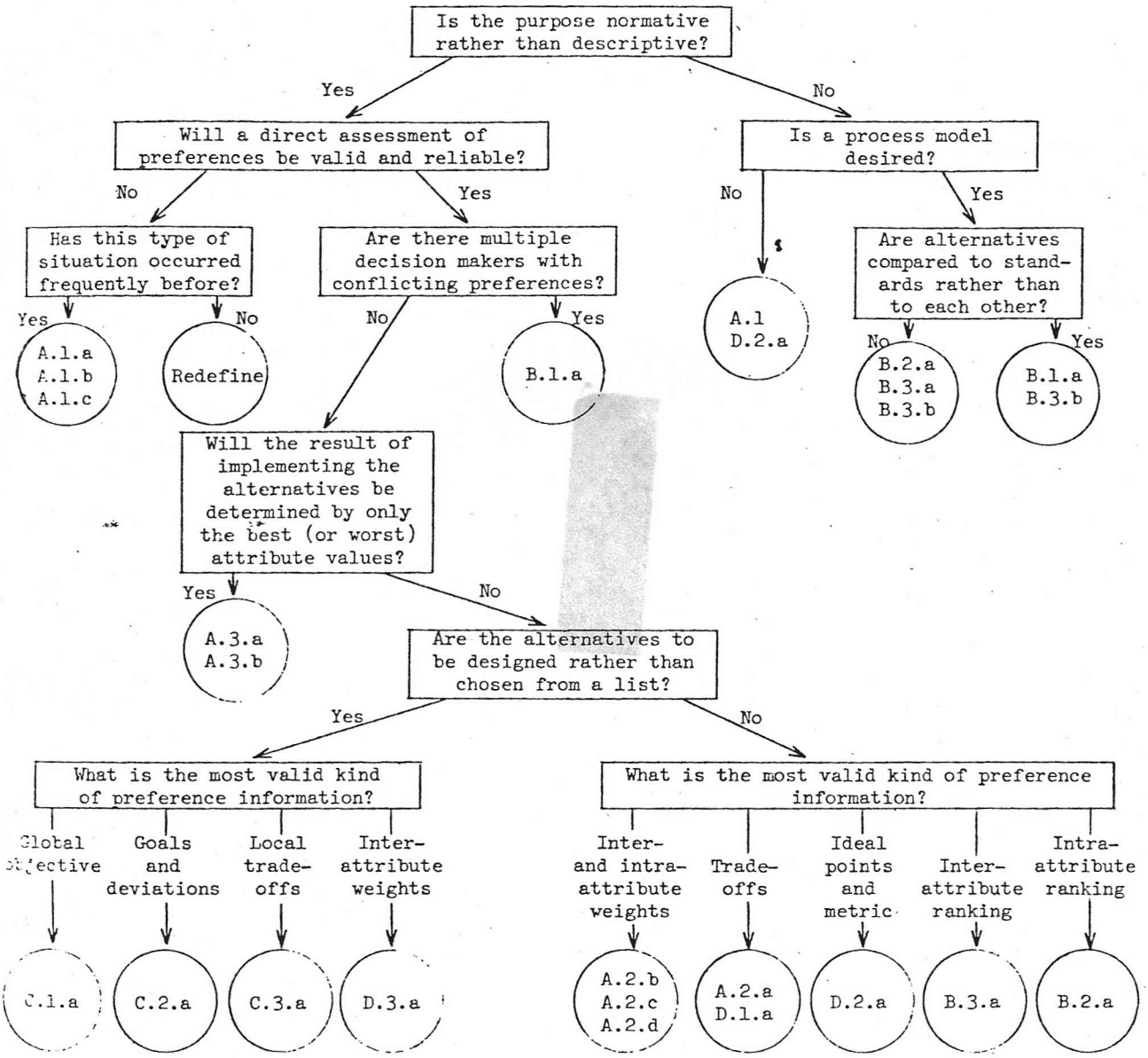


Figure 2 - A Taxonomy of Uses

I have thus brought forth, in essence, that the process of consulting is mapping from a conceptual world of situations and systems into a conceptual world of methods, establishing a link between the former part's concepts and the latter part's ones.

ENHANCING MODEL MANAGEMENT SYSTEMS

In a computerized model management system (MMS), methods are implemented to receive as input well specified and formatted data. Data base management systems (DBMS) are designed to handle retrieving, formatting, and organizing data files for models to be run, according to specifications. A MMS is a step above in sophistication in that it comes up with those specifications. It provides a user with information about the models and can perform tasks like descriptions, example displays, interfacing, etc.,and therefore extends and gives power to a DBMS when coupled to one. Ideally, a step higher towards the perfect MMS would be a system that could automatically structure a user's ideas, and shape them so as to give him/her a choice of appropriate models for solving his/her problem [8,9].

The question that naturally comes to mind now is: assuming that it is possible to build such a system, what tasks should it need to perform? Problem structuring is very difficult to any individual in a general world, and designing an all purpose intelligent computer system is still Dr. Frankenstein's dream. However, restricting our knowledge domain to an operations research and management science environment -- at least a well formalized subset of such an environment -- and structuring its concepts into a consistent epistemological formalism should provide a coherent ground to achieve a mapping between a situation to model and a satisfying mathematical model. Of course, it would take quite a deal of semantics and semantic processing highly conditioned by the consistency of the formalism underlying the knowledge domain. This domain being the Knowledge base through which to process from one end -- the user's concepts -- to the other -- the models' concepts. Creating and linking conceptual entities requires a very smart querying system whose inferencing power would rely on both structural consistency and heuristic processing.

It is towards designing this sort of computer system that I wish to aim this study. The attempt here is to propose potential answers to the following questions: (1) Is it possible to represent in a consistent, complete, and coherent way on a machine a knowledge domain so as to have the system access associations of concepts leading to deductions and inferences in a humanlike manner? (2) If the previous problem is solved, is it possible possible to program a machine to "intelligently" query a user based on the represented knowledge?

The first question is structural whereas the second is procedural, but both are purely epistemological. The dependence of the latter on the former gives a natural order of reflection. Indeed, before thinking of using some knowledge, it is necessary that the knowledge lie

somewhere. This explains why this study started focusing on researching an adequate structure by investigating the work done in the field of knowledge representation before prospecting what pertains to knowledge processing. The latter is a more difficult area whose state of understanding today does not go beyond structure manipulation. General inferencing algorithms are not available but routines particular to contextual domains -- idiosyncratic inferencing -- are sufficient. They will be written in terms of basic knowledge processing utilities.

A MMS doted with a problem structuring subsystem composed of a solid knowledge base and a "smart" processor is thus tremendously enhanced.

SYSTEM GENERAL DESCRIPTION

I would like to explain, as an overall image, what a decision support system with a concept processing facility could perform. Its basic functions are defining functions, validating functions, and checking functions. With such functions a user with a specific problem in mind would provide interactively inputs for three main phases: a problem structuring phase; a model validating phase; and a model running phase. As I detail the foregoing system, I shall try to pinpoint characteristics and issues that will determine criteria for the relevance of the knowledge representation presented in a later section.

1.0 BASIC FUNCTIONS

The tasks to be performed by the system are of three classes. Defining functions will deal with and within a conceptual knowledge base, defining concepts and instances pertaining to and characterizing the user's problem in his/her terms, "making some sense" in the domain context. Validating functions will ensure that what the system "understood" from the user's input is somehow what s/he meant. Finally, checking functions will make sure that everything is structurally sound and consistent.

1.1 Defining Functions

As the user interacts with the system s/he has conceptual entities in mind in terms of which s/he knows about his/her problem. Assuming that the knowledge base comprises these concepts the first task for the system is to create concepts and instances related to those concepts which constitute its "mind", in order to represent the user's entities and their explicit or implicit semantic import. This should be caught by the representational power of the knowledge base and how it carries all information that makes a concept be that concept. Attributes and roles they play within a concept, everything should have implicitly

access to a whole epistemological inheritance of concepts and properties. The defining functions will actually make use of the representation structure to guide the user.

To illustrate this let's think of a user dealing with a Production, Distribution, and Inventory (PDI) problem. There is production of goods at plants, then shipment of these goods to distribution centers, say. The model into which the system could try to map this situation would be a network of nodes and arcs, plants and distribution centers being nodes, and shipping routes being arcs. The system should then be capable of intelligently tracing a path of concept association to define an instance of the concept ARC with attributes FROM/NODE and TO/NODE to be SHIPPING with according attributes PLANT and DISTRIBUTION/CENTER filling the instantiated concept's attributes. As we shall see later, the Structured-Inheritance Network representation supports this capability, and more, and its associated language KLONE provides appropriate primitive functions with which to build the system's defining functions [2,3,4,5].

1.2 Validating Functions

Human knowledge is highly idiosyncratic. Conceptualizing the same entity is done quite differently for different persons. Thus, in an exchange between two persons, a constant validation process is present -- of course, unconsciously -- in order to make sure that both parties talk about the same thing. Because of the fact that at each conceptual stimulus much is implicitly meant at the same time, a necessary assumption making mechanism (in computer terms: default value assignment mechanism) is to be used in order to cut down the burden of asking recursively, and therefore indefinitely, for detailed conceptual inheritance. Hence, validation functions play the role of coming back to the user with assumed values when these need be activated in some fashion, to either confirm a previously made hypothesis, or adjust it when necessary. Such functions ought to be capable of maintaining conceptual consistency, and argue with the user upon encountering incoherent values, and resolve them.

For instance, taking the PDI context again, a user stating that s/he is shipping goods from Houston to Houston should trigger a reaction from the system. S/he would be answered something like "It does not make much sense to us to ship from a place to the same place; unless you could provide some clarification on how this could happen". Depending on what the user meant, an inconsistency may be resolved, or an input error may be corrected.

Validating functions, too, depend heavily on the representational power of the knowledge base. KLONE primitives, I believe, can support the principle.

1.3 Checking Functions

This last set of functions is more primitive than the previous ones, and are related to syntactic structural soundness of the "learned" knowledge created during interaction with the user. In this sense, they are not noticeable by the user and form a kind of compiler for the representation formalism. For example, a concept cannot have the same concept for superconcept and subconcept.

2.0 SYSTEM COMPONENTS

As I said earlier, the decision support system decomposes itself into three main stages: problem structuring, model validation, and model execution. This decomposition underlies the logical procedure of modelling.

2.1 Problem Structuring

As a user logs on a regular model management system, the interaction offered, although it may give the illusion of intelligent query, is essentially a hierarchical menu of commands and routines which is inevitably rigid, and preconditions in a deterministic way a template structure into which the user has to fit his/her problem. As a contrast, a system allowing one to define his/her own concepts and henceforth trying to tie them inferentially to those of some models would give a heuristic power that very much resembles a human consultant's way. The part of the system responsible for this task constitutes the problem structuring module. The context -- the application domain -- being common to both the user and the system, general concepts such as basic activities and typical processes involved, should bear enough information for the system to infer as much as possible, and also prompt for more stimuli when needed.

Works like the ones of Moore [11], and Collins, et al [7], explored the issue of reasoning from incomplete knowledge. The latter present a taxonomic approach to ways and circumstances of automatic inference processes in a computer aided instruction (CAI) environment based on a semantic network representation of memory [6]. The CAI context bears considerable resemblance with our problem structuring system in that the parties interacting learn from one another.

The problem of convergence of such a reciprocal learning process is the interest of the next module: the model validation.

2.2 Model Validation

It is hoped, in the process of structuring one's problem, that closure be eventually reached in achieving conversence towards some model, in a consistent way for both the user and the system. For this purpose, the system must have the capability of "making some sense" out of the user's input, and vice-versa. the user's concepts are expected to be found some analogous counterparts in the system's knowledge base. Therefore, a facility making sure that both agree on the analogy, make the same set of assumptions, and feel comfortable doing so, is necessary. Model validation tasks are principally checking for conceptual inconsistencies, resolving conceptual inconsistencies, exploring inferential consequences of hypotheses which may lead to conceptual inconsistencies, and finally, reach validation of previous defaulted assumptions. It is in the system design process that such modules are to be strategically set into a validation code.

There is no such thing to date -- to my knowledge -- as general inferring algorithms. Thus, situations will have to be well understood as major determinants of this phase, idiosyncratically if not universally. Hopefully, factors like the closure of the application knowledge base, and a thorough preliminary investigation of MMS usage, are likely to circumscribe a relatively small set of situations to be treated. An ongoing experiment towards building this sort of facility tends to give credit to this view.

2.3 Model Execution

The aim of the user is to find a mathematical decision model to run with his/her specific input. Therefore, the ultimate stage and the target of someone using the system is running some actual code.

Once conceptual connections are established between the user's problem and and adequate model, the latter ought to be fed data and run, automatically. Data organization specific to a model is interfaced via a classical DBMS from user specified sources.

At this stage, the user is comfortable with the problem modelling and is familiar with everything that must be provided for input in his/her terms, and will be given solutions in his/her terms also.

SEMANTIC NETWORKS

In this section, I present a brief summarizing survey of knowledge representation research done since the first appearances of semantic memory structures until today.

In the last twelve years, the issue of representing knowledge in a machine based on the idea of concept associations has been luring quite a number of creative scientists. The research has blossomed into a very fruitful number of representations of associative memory as a basis of model for a humanlike knowledge base. Since early and relatively simplistic models like Quillian's TLC [13], and Carbonell's SCHOLAR [6], artificial intelligence investigators, as much as psychologists, linguists, philosophers, and other obscure computer scientists of the "esoteric" side -- as opposed to the automatic theorem proving group of logicians constituting the formal hard core of artificial intelligence -- have formed a group of semantic network and language believers. Much is still going on, and the area promises to be a very fruitful and exciting one.

Linguists particularly, if not exclusively, were the ones to give the subject a close look during the early seventies -- since the birth of semantic networks is acknowledged to be 1966-68, with R. Quillian's pioneering work [12,13]. That resulted in the very used but rather chaotic (for reasons of rigidity) case structure in networks like those designed by Shapiro [17], Rieger [14], Simmons [18], etc. All these nets are based on centralizing an action on the verb whose agent (or subject) would but not necessarily commit the action (the verb) on a possible object. The common flaw in all the case structure representations is that the concept of concept is understood in a rather intuitive fashion as showed by Brachman in his dissertation [2].

In an excellent survey that has also the merit of being the recent and complete published in the area [4], he shows that everyone adopted a uniform convention: as concepts are entities subject to associations, they are simplisticly defined as nodal units interrelated with as many types of linkages as specifically needed. It thus appeared that what a concept was was not easy to perceive and even less to be put into a machine.

However, most of the experiments were somehow successful doing a somewhat intelligent work. But, except for non-network representations like Schank's [16], the structure of knowledge was very particularized to their experimental domains. If successful for the knowledge base and the procedures locally designed, the ambiguous, incomplete, or too large set of primitive nodes and links was very hard or impossible to translate into another domain of application. In fact, even Schank's representation had the notion of concept not clearly set.

P. Winston [19] was among the first to be concerned with epistemological bases for semantic networks. Also, as the seventies faded, the area expanded from nets to include languages like KRL [1], and FRL [15], for knowledge representation and manipulation, offering a complex data structure immediately codable as a network knowledge base.

Unfortunately, the difficulty of defining a standard, well understood grammar for semantic networks persisted. The reason was that the notion of concept varied wildly with the network designer's intuition. Not before a paper by Woods [20] was published in 1975, was this question tackled as such. What a concept is, how to capture its

internal attributes tied by properties, examples and instances, associations, connotations of meanings, deductions and intelligent inferencing, and what not, into a well defined, complete, and minimal apparatus of logics, is the challenge. In his PhD thesis at Harvard, R. Brachman came up with a network formalism supporting parts of answers to this challenge: the so-called Structured-Inheritance Networks (SI-Nets).

KNOWLEDGE REPRESENTATION USING SI-NETS

The purpose of this section is to summarize the basics of R. Brachman's representation formalism which I think is powerful and explicit enough so as to make a system of the kind we described above be practically envisageable. For more details and in-depth discussion on the material presented here, refer to [2,3].

Brachman distinguishes two types of basic concepts: a generic concept type defined as a prototypical object whose description of parts and structure characterizes eventual individual objects so as to make them be instances of the concept they individuate; and an individual concept type which is the one of the instances of generic concepts. A third virtual concept type is a parametric individual concept type which is used in the descriptive part of a generic concept as a copy of some other generic concept.

A generic concept (represented as an oval node) is a set of attributive parts defining roles within this concept. Generic roles, thus called because they are attached to a generic concept (represented by a square node), are internal parts of a concept made up of so-called role facets. These facets identify a role and are:

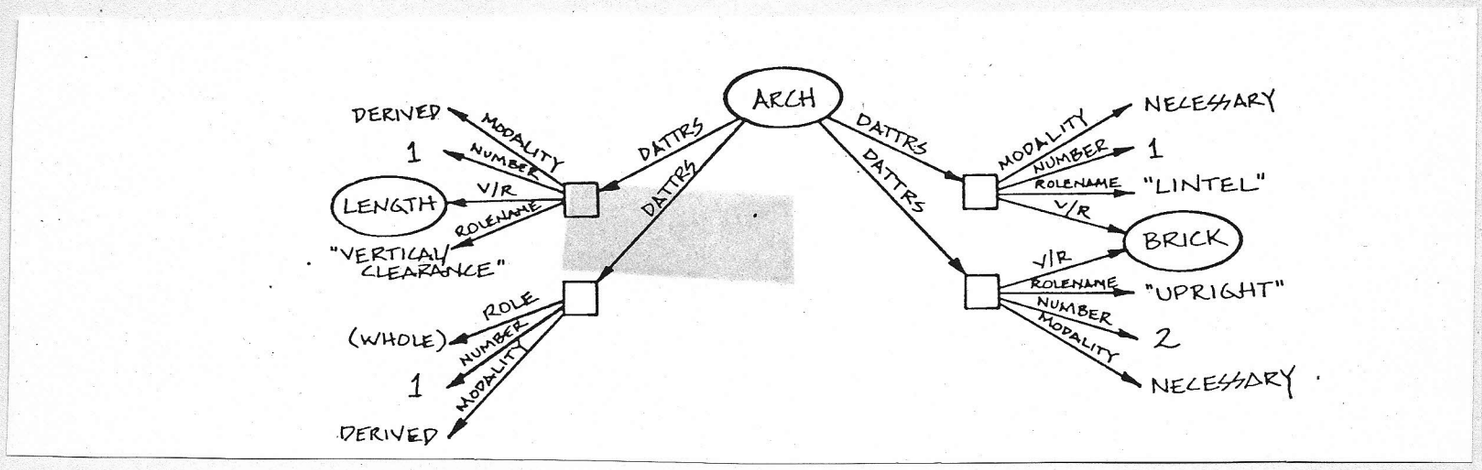
ROLENAME: a mnemonic label for the role played by the attribute in the defined concept; e.g., TAIL in the concept ALLIGATOR.

VALUE/RESTRICTION: a generic concept indicating the sort of entity with which the role is expected to be filled in a individual instance of the concept.

MODALITY: stating that the attribute is Necessary, Derived, or Optional.

NUMBER: a predicate telling how many fillers are allowed for the attribute.

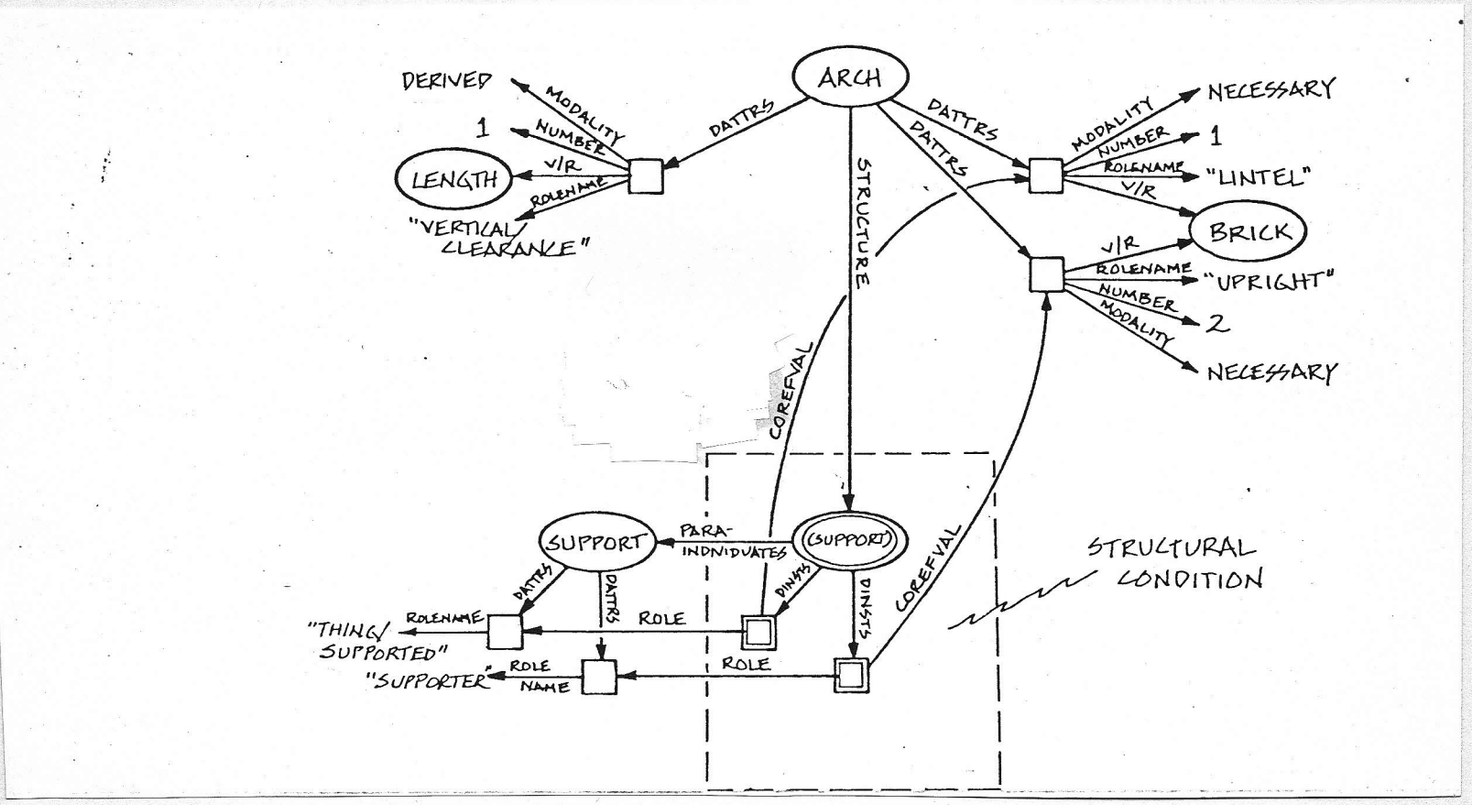
Figure 3 gives an example of a generic concept: ARCH is two UPRIGHTs and a LINTEL.



Fisure 3 - A SI-Net Concept

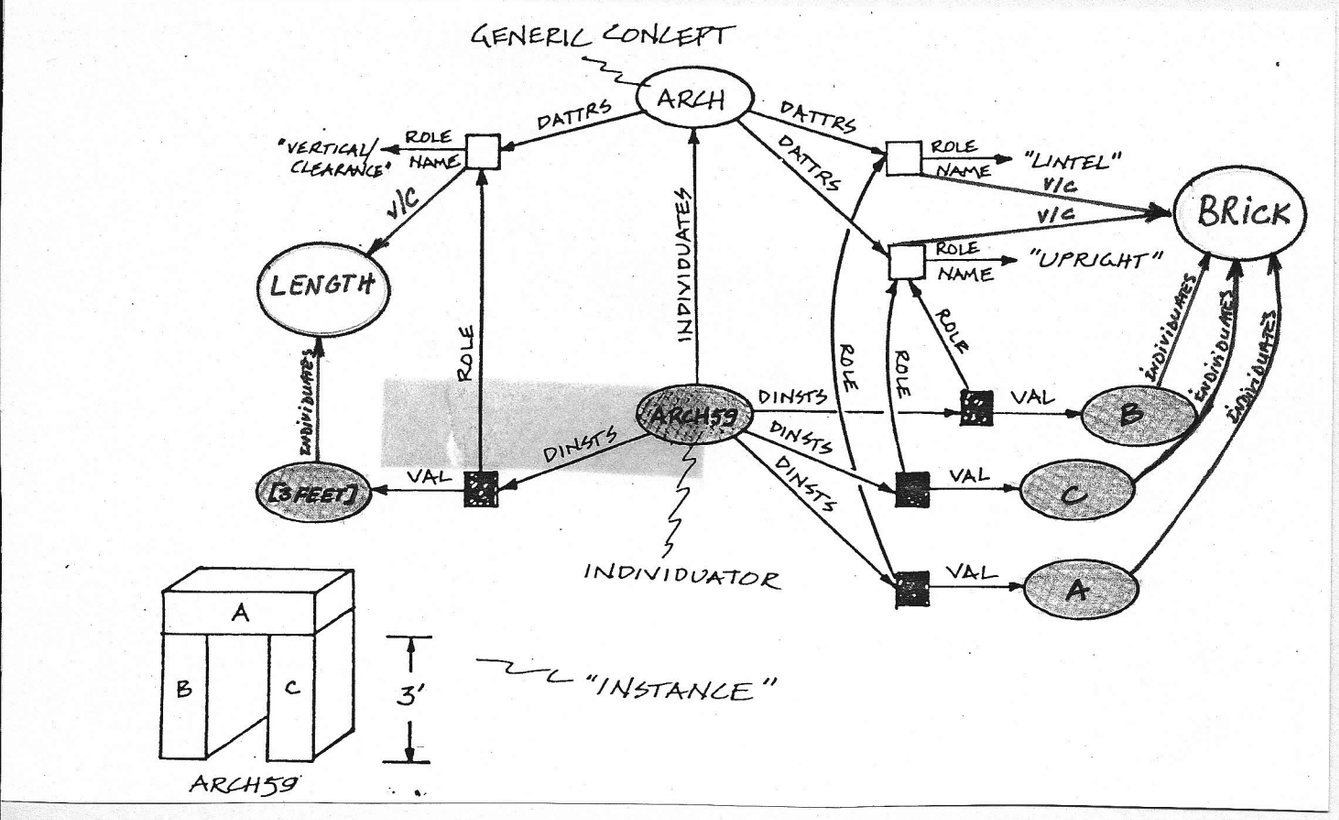
A seneric concept is not completely defined without describins how its roles so tosether; e.s., what makes the difference between a pile of bricks and a stack of the same bricks. The structural condition (S/C) takes care of this. Very much like a dictionary defines a word in terms of other words, a S/C describes a concept in terms of other concepts. To the precision that the concepts used in a S/C are virtual copies of concepts instead of those concepts themselves, as they are instances parameterized by the concept to which description they contribute. They are thus called Parametric Individual Concepts (PIC), or Paraindividuals, (represented by double oval nodes) and their roles (represented by double square nodes) corresponding to the roles in the concept they "paraindividuate" coreference the roles of the described concept. This is why they are called "corefroles".

Fisure 4 illustrates a S/C for the concept ARCH statins that the UPRIGHTs support the LINTEL.



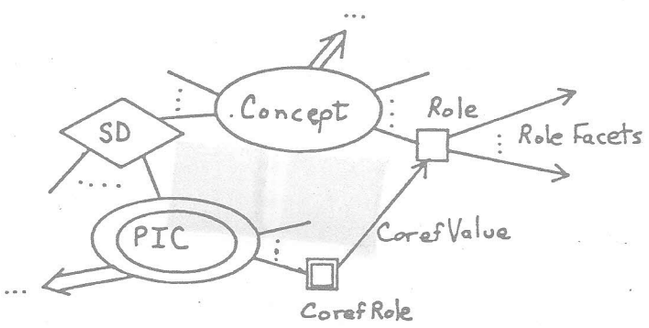
Fisure 4 - A Structured Concept

An individual concept (represented by a shaded oval node) is a particular instance of a seneric concept. Therefore, it satisfies the requirements of each of the seneric roles' facets with its instance roles (represented by shaded square nodes). In particular, the number of satisfiers of each seneric role must asree with the number facet fisurins there; and the value must be an individual instance of the value/restriction, as shown in fisure 5.



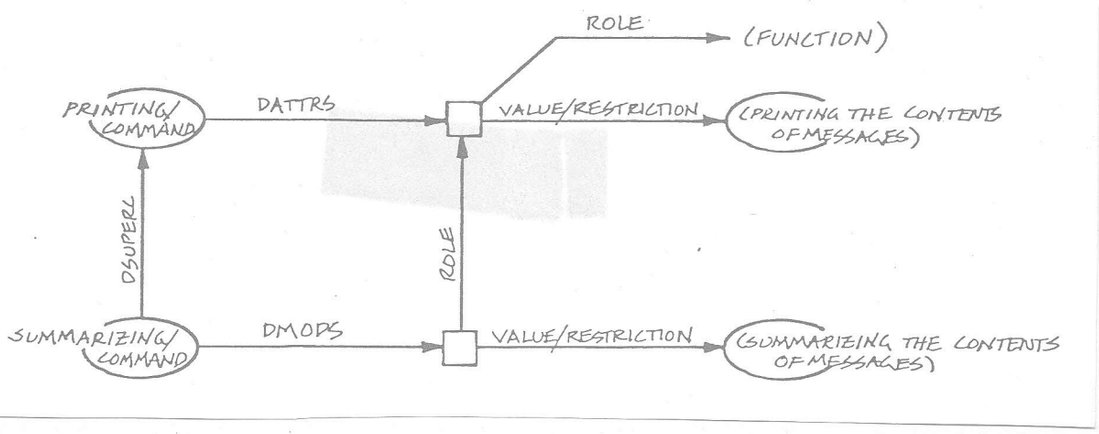
Fisure 5 - An Instance Concept

To sum up, the diasram below (fisure 6) illustrates the internal structure of a seneric concept.

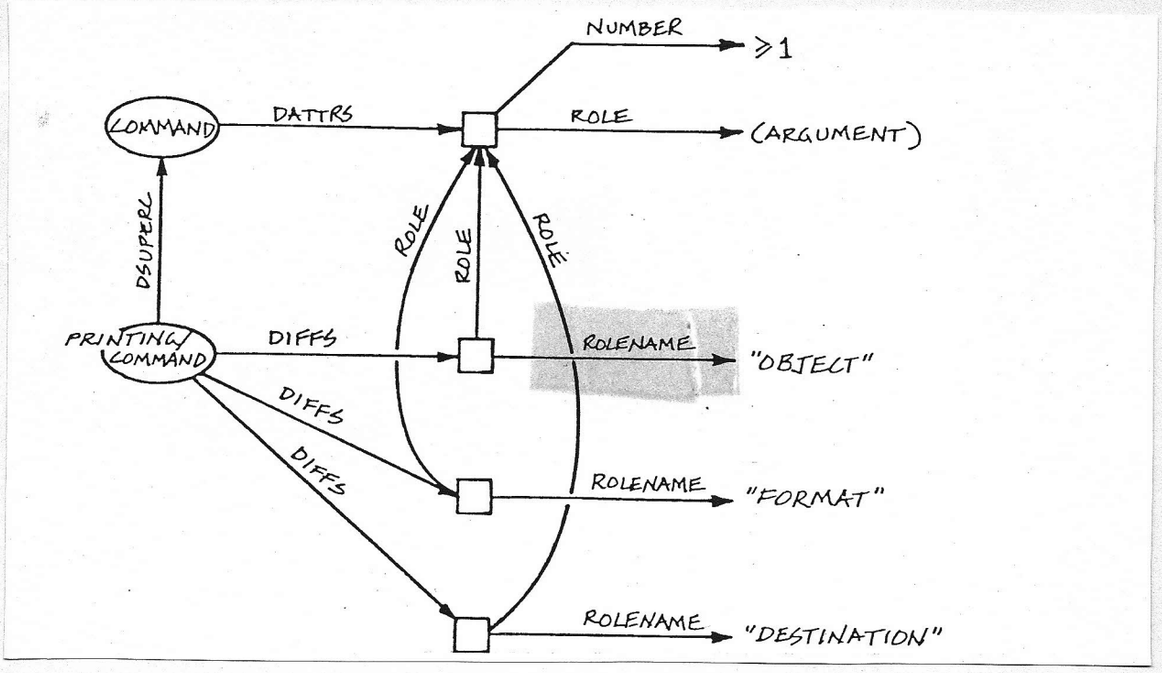


Fisure 6 - Typical Internal Structure of a SI-Net Concept

Inter concept relations, besides individuation and paraindividuation are essentially subconcept-superconcept relations. Roles are inherited intact from superconcepts in a subconcept, except in two cases: particularization and differentiation of roles. The former applies when some role facet(s) of the subconcept (also called specialized concept) are modified (see fisure 7), and the latter happens when a role needs to be broken into several subroles (see fisure 8).



Fisure 7 - Modified Role Inheritance



Fisure 8 - Differentiated Role Inheritance

CONCLUSION

The ideas exposed in this paper would benefit a lot from a practical application. They are presently used in the design of a model management system to be intesrated in a decision support system: the DAISY system [9], at the Decision Sciences department of the Wharton School. I also intend to further investisate the desisn of an automatic consultins system and try to build an experimental prototype to fulfill my doctoral research requirement.

Further research needs to be done principally in processins knowledse, comins up with a set of deductive and inferencins alsorithms. Retrieval, buildins, and predicative primitives associated to the SI-Net formalism have been set up at B.B.N. as KLONE [5], a knowledse manipulatins lansuase compatible with the structure presented. KLONE utilities will most likely constitute the primitives with which to build the more challensins alsorithms mentioned above.

REFERENCES

[1] Bobrow, D., and T. Winograd, "An Overview of KRL, a Knowledge Representation Language", COGNITIVE SCIENCE, Vol.1, No.1, January 1977, pp. 3-46.

[2] Brachman, R., "A Structural Paradigm For Representing Knowledge", BBN Report 3605, Bolt Beranek and Newman, Inc., Cambridge, Mass., (1978).

[3] _____, "Theoretical Studies In Natural Language Understanding", BBN Annual Report 1 May 1977 - 30 April 1978, BBN Report 3888, Bolt Beranek and Newman, Inc., Cambridge, Mass., (1978).

[4] _____, "On the Epistemological Status of Semantic Networks", in ASSOCIATIVE NETWORKS -- THE REPRESENTATION AND USE OF KNOWLEDGE BY A COMPUTER, N.V. Findler, Ed., NewYork: Academic Press, (1979).

[5] Brachman, R., ET AL, "KLONE Reference Manual", BBN Report 3848, Bolt Beranek and Newman, Inc., Cambridge, Mass., (1978).

[6] Carbonell, J., "AI in CAI: an Artificial Intelligence Approach to Computer-Aided Instruction", IEEE TRANSACTIONS ON MAN-MACHINE SYSTEMS, MMS-11, 1970, pp. 190-202.

[7] Collins, ET AL, "Reasoning from Incomplete Knowledge", in REPRESENTATION AND UNDERSTANDING, D. Bobrow and A. Collins, Eds., NewYork: Academic Press, (1975).

[8] Elam, J., "Model Management Systems: an Overview", Working Paper, Department of Decision Sciences, The Wharton School, University of Pennsylvania, (1979).

[9] Hurst, G., ET AL, "DAISY: a Decision-Aiding Information System", Working Paper, Department of Decision Sciences, The Wharton School, University of Pennsylvania, (1975).

[10] McCrimmon, K., "An Overview of Multiple Objective Decision Making", in MULTIPLE CRITERIA DECISION MAKING, J. Cochrane and M. Zeleny, Eds., University of S. Carolina Press: Columbia, SC,(1976).

[11] Moore, R., "Reasoning from Incomplete Knowledge in a Procedural Deduction System", MIT AI-TR-347, December 1975.

[12] Quillian, R., "Semantic Memory", in SEMANTIC INFORMATION PROCESSING, M. Minsky, Ed., Cambridge: MIT Press, (1968).

[13] _____, "The Teachable Language Comprehender", COMMUNICATIONS OF THE ACM, Vol. 12, 1969, pp. 459-475.

[14] Rieger, C., "An Organization of Knowledge for Problem Solving and

Language Comprehension", ARTIFICIAL INTELLIGENCE, Vol. 7, No.2, Summer 1976, pp. 89-127.

[15] Roberts, R., and I. Goldstein, "FRL User's Manual", AI-Memo No. 408, MIT A.I. Lab., April 1977.

[16] Schank, R., "The Structure of Episodes in Memory", in REPRESENTATION AND UNDERSTANDING, D. Bobrow and A. Collins, Eds., NewYork: Academic Press (1975).

[17] Shapiro, S., "A Net Structure for Semantic Information Storage, Deduction, and Retrieval", in PROCEEDINGS OF THE 2ND IJCAI, 1971, pp. 512-523.

[18] Simmons, R., "Semantic Networks: Their Computation and Use for Understanding English Sentences", in COMPUTER MODELS OF THOUGHT AND LANGUAGE, R. Schank and K. Colby, Eds., San Francisco : W.H.Freeman and Co., (1973), pp. 63-113.

[19] Winston, P., "Learning Structural Descriptions from Examples", Project MAC TR-76, MIT, Cambridge, Mass., (1970).

[20] Woods, W., "What's in a Link: Foundations for Semantic Networks", in REPRESENTATION AND UNDERSTANDING: STUDIES IN COGNITIVE SCIENCE, D. Bobrow and A. Collins, Eds.,NewYork: Academic Press, (1975).