The Procedural Semantic Network: An Extensible Knowledge Representation Scheme

Peter F. Patel-Schneider, Bryan M. Kramer, Andrew Gullen,
Yves Lesperance, Hugh Meyers, John Mylopoulos

Department of Computer Science
University of Toronto
Toronto, Ontario

Abstract

The Procedural Semantic Network (PSN) is an extensible knowledge representation scheme[1] with several unusual features and unique design criteria. This paper outlines the current state of PSN emphasizing its adherence to the design criteria of uniformity and self-description. It also describes the current implementation of PSN at the University of Toronto and summarizes research on PSN being conducted here.

1. Introduction

PSN is a knowledge representation scheme (hereafter KR scheme) based on semantic network notions such as those of object, link, and IS-A. However, instead of requiring a global interpreter, a PSN knowledge base (KB) contains procedures attached to each generic object (or class) which have the very important role of determining the instance and inheritance relationships for that class along the lines of the frame proposal [Minsky 75]. Further, PSN has a very strict notion of inheritance in its IS-A hierarchy as opposed to many other KR schemes which allow most inherited information to be overridden.

PSN consists of a kernel and various extensions which together form a set of KR schemes ranging from the very basic to the quite complex. These various KR schemes allow a user to select a level of functionality that is appropriate for his application. He does not need to include unwanted functionality with its extra cost and complexity. PSN is designed so that further or different extensions can be built from existing features and integrated into it, thus accomodating applications that do not fit well into current PSN. If this was not possible an ad-hoc extension (one that cannot be integrated into PSN and would likely be useless in unrelated application areas) would be required or a totally new KR scheme would have to be designed. Both of these routes lead to a proliferation of incompatible KR schemes and make it harder for users to select an appropriate KR scheme.

PSN adheres to two very important design criteria throughout all its extensions. The first is the uniformity of a KR scheme. A scheme is uniform if it has only a few basic concepts and all features in it are naturally applicable over large, regular sets (ie. with few exceptions). The Actor formalism [Hewitt 73] and Smalltalk [Ingalls 78] are highly uniform schemes as all entities within a program are of one basic kind. Uniformity is a powerful design criterion in that it discourages the "kitchen sink" syndrome where new constructs are added for each new concept being represented without considering the interaction of these new concepts with the rest of the formalism. This help is particularly important in a highly extensible KR scheme such as PSN.

The second design consideration is self-description: the degree to which the

[1]: Following Hayes [Hayes 74] we define a knowledge representation scheme as a formal notation for representing knowledge.

meanings of different features of a KR scheme are representable and accessible within the scheme itself. LISP, for example, gains much power from its high degree of self-description. This criteria has a direct impact on the ease with which a scheme can be extended and which the extensions can be understood.

Together these two design criteria give PSN a coherence that many extensible systems lack. The next section of this paper gives an overview of PSN stressing the importance of these two design criteria in an extensible KR scheme.

A new implementation of PSN is currently in progress at the University of Toronto. This LISP implementation follows the extensible design of PSN without paying for the usual cost of self-descriptive systems and will become available to users not closely connected with the PSN project. The third section of this paper discusses the current implementation effort. Finally we give an overview of research questions related to PSN that are currently being considered at the University of Toronto.

2. The Extensible Design of PSN

PSN currently has six extensions in addition to its kernel. The extensions are generally additive in that each one builds on the previous ones. This section will introduce the basic ideas and features of the kernel and each extension with emphasis on how the main PSN design criteria impact on their design. These short discussions, although they suppress detail, serve to illustrate the basic design philosophy of PSN and demonstrate how adherence to the design philosophies of uniformity and self-description serve to mould PSN.

The kernel of PSN is the basis of all extensions. The kernel has been greatly influenced by Abrial [Abrial 74] and adheres to the following design criteria: non-redundancy -- no feature in the kernel is naturally or directly expressible in terms of the other features, minimality -- no unnecessary features exist in the kernel, completeness -- the kernel is sufficient for the definition of any feature in the extensions, and extensibility -- the kernel is naturally extensible to include arbitrary new features.

The basic unit of a PSN knowledge base (KB) is the object which represents either an entity in the world being modelled or a useful concept in the organization of the KB. The most basic uniformity constraint in PSN is that everything that is represented by a PSN KB is represented as an object and can therefore be manipulated as an object. A class, in PSN, is a particular type of PSN object which represents a generic concept and has other objects as instances. Another very important uniformity constraint in PSN is that every object in a PSN KB must be an instance of a class, in particular every PSN KB has the classes 'OBJECT', which has as instances all objects, and 'CLASS', which has as instances all classes. Note that 'OBJECT' is an object and 'CLASS' is a class and thus they are both instances of themselves.

What it means to be an instance of a class is determined by four procedures attached to the class. One of these procedures is responsible for adding instances to the class, one for removing instances from the class, one for determining whether an object is an instance of the class, and one for generating all instances of the class. These four procedures suffice to define the semantics of each class: this differs from most KR schemes in which an underlying and unchangeable interpreter, which cannot be manipulated from within the scheme, determines the semantics of all construcs. In particular the four procedures attached to each of 'OBJECT' and 'CLASS' define the semantics of these very important classes. These procedures are, more than any others, responsible for the behaviour of the PSN kernel. Their inclusion in PSN allows the behaviour of PSN to be investigated and modified within PSN and forms the basis of

the self-descriptive aspects of PSN.

Some classes in the PSN kernel have as instances objects which represent binary relationships between objects. Each of these classes thus forms a binary relation from objects to objects. All of these classes are instances of the meta-class 'RELATION' whose four procedures are responsible for the behaviour of relations. Finally, the class 'PROCEDURE' is part of all PSN KBs and is responsible for giving meaning to procedures which, of course, are objects and must be grouped into a class.

Part of a small PSN kernel knowledge base (without most of the attached procedures and several instance relationships) is shown declaratively in Figure 1. The figure represents two instances of the class 'PERSON', namely 'John' and 'Mary, two relations, 'BROTHER' and 'SEX', and the appropriate relationships for 'John' and 'Mary' in these relations. The four classes 'OBJECT', 'CLASS', 'RELATION', and 'PROCEDURE' form a prominent role in this KB, as in all PSN KBs.

The figure hides the fact that to do anything useful in the PSN kernel a KB designer has to be concerned with the procedures that support classes and relations. For example, to enforce constraints such as "for each instance of 'PERSON' there must be an instance of 'SEX' mapping it to a sex" the add procedure for 'PERSON' must be explicitly modified by the KB designer.

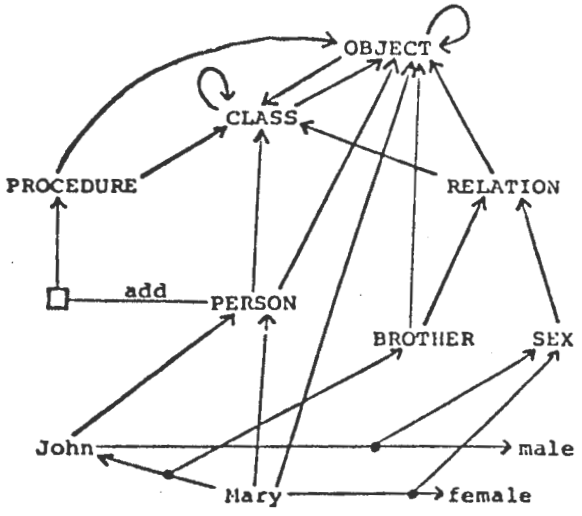


Figure 1

The PSN kernel defines the notion of object, the notion of class with attached procedures defining the instances of each class and the behaviour of the class, the notions of relation and procedure, and the four predefined classes which hold things together. Here in the kernel are the most important uses of uniformity, the constraints that everything must be an object and that all objects must be instances of classes that describe the behaviour of their instances through attached procedures. This uniform basis allows many different sorts of extensions to be made and also allows the extensions to be wide-ranging. Thus an extended version of PSN should also be uniform and will encourage further extensions. The kernel also lays down the basis of self-description in PSN by its inclusion of the objects 'OBJECT', 'CLASS', 'RELATION', and 'PROCEDURE'. These objects have information attached to them in the form of procedures that describe the behaviour of all other objects and a major part of the extensions to PSN are extensions to these procedures.

The first extension to the kernel is the addition of an IS-A hierarchy. The basic idea of this partial-order hierarchy is that if class A IS-A class B then every instance of A is an instance of B (ie. this IS-A hierarchy has no exceptions). This extension is effected by defining a new relation called 'IS-A'. The four procedures attached to 'IS-A' ensure that it forms a partial order on classes. To complete the implementation of this extension the procedures attached to the objects defined by the kernel must also be extended to ensure that the IS-A hierarchy has the intended meaning. This involves creating 'IS-A' links as

appropriate and ensuring that objects are added to IS-A parents when added to a class. All these extensions to procedures do not alter PSN's behaviour when there are no IS-A links and thus are truly extensions.

The IS-A extension can be considered in one of two ways: as the definition of a new relation and various supporting procedures along with extensions to several other procedures or as the addition of a strict IS-A hierarchy to PSN. A KB designer (ie. a typical PSN user) should only consider the second interpretation of this extension and the implementation of the new concept should remain hidden from him. However, the implementation of an extension is useful when investigating its behaviour or when adding further extensions and forms part of the self-descriptive nature of PSN. This dualism between intended meaning and implementation is present in all PSN extensions.

The second extension to PSN is an independent enhancement to the kernel introducing slots [Minsky 75] (alias roles [Brachman 79], etc.). Each class in this extension to PSN can define slots which are functional relationships between the class and other classes. That is, each slot of a class maps each instance of the class to a corresponding value. Slots are introduced as special relations which are also instances of the class 'SLOT' defined by this extension which constrains the values of slots to be unchangeable. As in the case of the IS-A extension, the procedures attached to 'CLASS' and 'OBJECT' must be extended to correctly handle the set-up and use of slots. The net effect of this extension is to introduce a method for aggregating information about an object into one conceptual unit. As with other extensions, the self-descriptive aspects are hidden from normal users.

These first two extensions can be combined together and then further extended. This third extension involves the inheritance of slots down the IS-A hierarchy. Since an instance of A is an instance of all A's IS-A ancestors, A can be said to have inherited all of the slots of its ancestors. We extend this by allowing the definition of A to restrict, but not otherwise modify, the possible values for slots that it inherits. This restriction of slots is effected by the creation of a new slot which is related to the old via a 'RESTRICTION' link (similar to a DMODS link in KL-ONE). This again involves modifications of the predefined procedures. This extension does not introduce much in the way of new concepts to PSN but is largely concerned with extending the reach of existing concepts or broadening their applicability and usefulness. In this way it increases the uniformity of PSN and shows how the basic design criteria of PSN provide guidance not only on how to proceed with an extension but also help to dictate what its content should be.

With these three extensions PSN has an IS-A hierarchy plus an integrated slot mechanism both having strict inheritance rules. At this stage several declarative features can be added to PSN to hide some of the procedural aspects of the kernel. This results in basic PSN as shown by the small knowledge base in Figure 2. Two classes, 'PERSON' and 'STUDENT' with 'STUDENT'

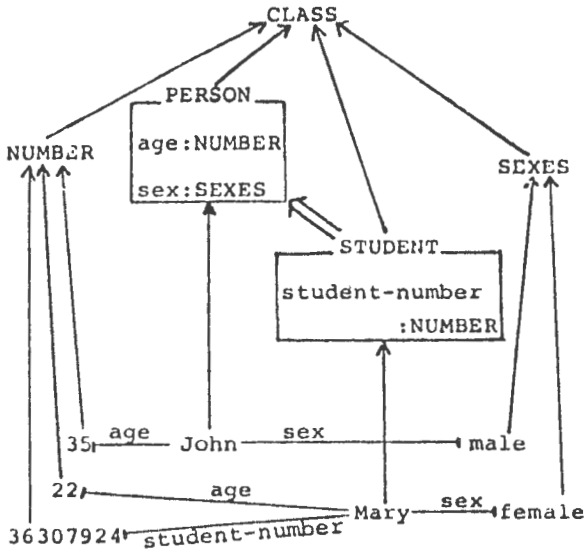


Figure 2

an IS-A descendant of 'PERSON' are shown in this figure. Also shown are definitions of the slots 'age', 'sex', and 'student number' and their values for 'John' and 'Mary'. This knowledge base shows the additions of IS-A and slots in a declarative manner, the way that most users would view them. These users would not care that these features are effected via extensions of the procedures attached to the predefined objects of the kernel and to them basic PSN would have a large declarative core.

However, the derivation of basic PSN from the PSN kernel is available to those who wish to investigate the procedural behaviour of the extensions or to further extend these concepts (as we have already done). Basic PSN is still highly uniform (the extensions apply over all classes uniformly) and has considerable self-description (as shown by the derivation of the extensions) and thus can itself be easily extended.

The remaining extensions to PSN all build upon basic PSN. The first of these extensions generalizes the slot mechanism already introduced [Kramer 80b]. The basic idea of this extension is to constrain all properties (both slots and slot definitions) of an object by means of properties attached to the classes of which the object is an instance. (Previously the slot values of an object were constrained by the slot definitions in the classes to which it belongs but slot definitions were unconstrained.) This new mechanism allows the user to impose constraints on the definitions of slots by means of special slots called meta-slots (and also on these constraints (and so on)). For example, a meta-slot can constrain the number of slots of a given type in a class. With this extension the user can himself add new types of slot definitions that have their own constraint mechanism. PSN therefore becomes more self-descriptive because one of its constraint mechanisms, slots, is now described within PSN. Uniformity within PSN is also increased because the special properties of 'SLOT' are replaced by a hierarchy analogous to the instance hierarchy for objects (consisting of objects, classes, meta-classes, etc.). Thus uniformity and self-description not only allow this extension to be made but strongly suggest that it be made and how it should be designed.

The next extension is concerned with making procedures examinable and thus more like other objects. In this extension procedures are represented as classes. Each procedure has various slots attached to it which include the operations of the procedure. Since procedures are now classes with slots they can be usefully placed in the IS-A hierarchy. The IS-A inheritance of slots can now be used to inherit the procedural information enclosed in the slots of procedures and thus a procedure can be made more specialized by modifying individual slots or adding new slots instead of modifying the entire procedure. This allows refinements to existing procedures to be created more easily [Borgida et al 82], an important consideration when creating the attached procedures for classes in a large IS-A hierarchy.

The sixth extension currently in PSN creates a class of relationships between classes [Lesperance 80a]. One of these relationships is the IS-A relationship and each of them is a method of organizing knowledge about classes. This generalization of IS-A allows the representation of non-strict inheritance such as similarity mappings and exceptional classes, thus increasing the utility of PSN. It also provides a meta-description for IS-A thus enhancing the self-descriptive aspects of PSN.

A small knowledge base showing some of these extensions in a declarative manner is given in Figure 3. This figure actually shows parts of two procedures demonstrating their declarative slots and the meta-slots defined in 'PROCEDURE' that constrain them and define their behaviour.

With these final extensions PSN achieves a complexity similar to other KR schemes (such as

KL-ONE [Brachman 79] and KRL [Bobrow and Winograd 77]). However, if all this complexity is not needed then all the extensions do not have to be used. This allows users who want a simpler or smaller system to co-exist with those that want a complex system offering many features for organizing knowledge.

In each extension to PSN the design criteria of uniformity and self-description are used in two complementary ways. First, they contribute to the ease of extending PSN. Second, they provide a measure by which the suitability of a proposed extension can be measured: if an extension does not retain or enhance uniformity or if it does not lend itself to self-description then it is not a desirable extension to PSN. In this way PSN and its extensions are forced to form an integrated whole. Any further extensions to PSN must follow this standard.

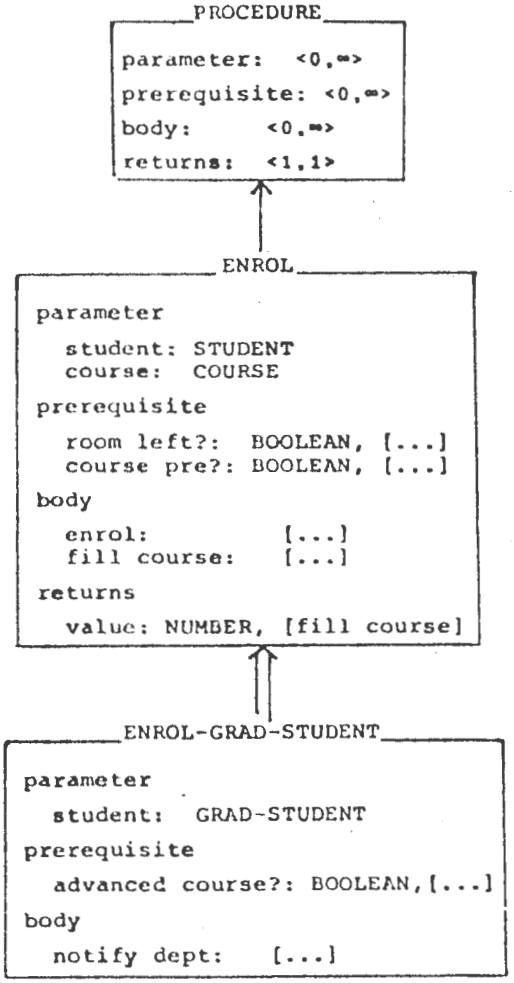


Figure 3

3. The Implementation of PSN

The current LISP implementation of PSN at the University of Toronto attempts to mirror the extensible aspects of PSN without paying the very high penalty in terms of efficiency that a direct implementation of the self-descriptive aspects of PSN would. A previous implementation of PSN and most of its extensions plus an interface language [Kramer 80] resulted in a very large program and has been put aside due to insufficient computing resources. The goal of efficiency in the new implementation is partly a result of this experience.

The idea behind the implementation is to have a layer corresponding to each useful set of extensions in PSN. These layers would not be self-descriptive as PSN is because the procedural extensions would be directly implemented in LISP and not in PSN itself. This means that the implementation of an extension would not necessarily be directly accessible and modifiable from within PSN but would produce a much more efficient system. However, aside from this, the implementation will have the same behaviour as a direct implementation. This is much the same as the difference between interpreted and compiled LISP, where the PSN implementation is essentially a hand compiled and optimized version of PSN.

A critical problem with such a layered implementation is that an application written in one layer may not work in another. This could arise from incompatibility of internal structure or from reliance on some code particular to one layer. We plan to retain as much upward and

downward compatibility as possible in the implementation. If an application is originally written in one layer of the implementation then it will work in a layer with more extensions and if it does not use some of the extensions present in a layer then it will work in a layer not containing those extensions. This is precisely what would be expected if a direct implementation of the extensions was done.

Currently only two implemented layers of PSN are available: PSN/0, an implementation of the kernel, and PSN/1, an implementation of basic PSN. These two layers are faithful implementations with only minor differences from PSN. A third layer, incorporating some of the extensions after basic PSN an probably several other extensions, is in the process of development.

The implementation is expected to give another impetus to the development of PSN as applications are developed and users determine what are the most useful features of PSN. We expect that the adherence of PSN to the design criteria of uniformity and self-description will help it resist the poor compromises that are often forced on KR schemes by users who want a particular feature implemented without considering how it will impact on the KR scheme as a whole.

4. Research Directions and Applications

Although PSN has had influence on the development of several knowledge-based systems (e.g. [Cohen 78], [Tsotsos 80]), it hasn't been used yet as a knowledge representation scheme in its own right. However, the current implementation is intended to be used by a number of projects involving the development of knowledge-based systems for medical applications (e.g. ALVEN [Tsotsos 80], CAA [Shibahara et al 82]). In fact, the features of PSN to be included in PSN/2 were determined in part by the knowledge representation needs of the group that intends to use it.

Many research questions have been raised from our work on the PSN project. The representation of events along with causal and temporal links relating them appears to be an important issue, particularly for a KR scheme that is to be used in medical applications such as the CAA system. We would also like to formalize the notion of KR scheme extensibility discussed in this paper to make precise statements about the compatibility of KBs developed at different PSN layers. A third research question involves the development of an object-based representation for assertions. We have already done much work on such a representation for procedures so that PSN procedures have all the features of classes. We would like to do something similar with assertions so that a PSN KB can include entities, procedures, and assertions all represented within a single object-based framework.

5. Bibliography

[Abrial 74] "Data Semantics" in Data Management Systems, ed. Klimbie and Koffeman. North-Holland, 1974.

[Bobrow and Winograd 77] Bobrow, Daniel and Terry Winograd. "An Overview of KRL, a Knowledge Representation Language". Cognitive Science I, 1, January 1977.

[Borgida et al 82] Borgida, Alex, John Mylopoulos, Harry K. T. Wong. "Taxonomic Software Development", in Three Perspectives in Conceptual Modelling, ed. M. Brodie et al. Springer-Verlag, 1982.

[Brachman 79] Brachman, Ronald J. "On the Epistemological Status of Semantic Networks", in Associative Networks: Representation and use of knowledge by computers, ed. Nicholas V. Findler. New York: Associated Press, 1979.

[Brachman and Smith 80] Brachman, Ronald J. and Brian Smith, eds. Special Issue on Knowledge Representation. SIGART, No. 70, February 1980.

[Carbonell 80] Carbonell, Jaime G. "Default Reasoning and Inheritance Mechanisms on Type Hierarchies". Proceedings of the Workshop on Data Abstraction, Databases, and Conceptual Modelling, Pingre Park, Colorado, 1980, SIGART No. 74, January 1981.

[Cohen 78] Cohen, Philip. On Knowing What to Say: Planning speech acts. Tech. Report No. 118, Dept. of Computer Science, University of Toronto, 1978.

[Dahl and Hoare 72] Dahl, O. J. and C. A. R. Hoare. "Hierarchic Program Structures" in Structured Programming, ed. Dahl, Dijkstra, and Hoare. Academic Press, 1972.

[Davis 77] Davis, R. "Interactive Transfer of Expertise: Acquistion of New Inference Rules". Proceedings IJCAI-77, Boston, 1977.

[Freeman and Leitner 80] Freeman, M. and H. Leitner. "Structured Inheritance Networks and Natural Language Understanding". Proceedings IJCAI-79, Tokyo, 1979.

[Goldman and Wile 80] Goldman, N. and D. Wile. "A Database Foundation for Process Specification". ISI/RR-80-84, Information Science Institute, 1980.

[Hayes 74] Hayes, Patrick J. "Some Issues and Non-Issues in Representation Theory". Proceedings AISB Summer Conference, University of Sussex, England, 1974.

[Hendrix 79] Hendrix, Gary G. "Encoding Knowledge in Partitioned Networks", in Associative Networks: Representation and use of knowledge by computers, ed. Nicholas V. Findler. New York: Associated Press, 1979.

[Hewitt 73] Hewitt, Carl. "A Universal ACTOR Formalism for Artificial Intelligence". Proceedings IJCAI-73, Stanford University, California, 1973.

[Ingalls 78] Ingalls, D. H. "The Smalltalk-76 Programming System: Design and Implementation". Proceedings 5th Annual ACM Symposium of Principles of Programming Languages, Tucson, Arizona, 1978.

[Kramer 80a] Kramer, Bryan. The Representation of Programs in the Procedural Semantic Network Formalism. Tech. Report No. 139/80, Dept. of Comp. Science, University of Toronto, 1980.

[Kramer 80b] Kramer, Bryan. "Representing Programs in PSN". Proceedings 3rd CSCSI/SCEIO Conference, University of Victoria, British Columbia, 1980.

[Lesperance 80a] Lesperance, Yves. "Handling Exceptional Conditions in PSN". Proceedings 3rd CSCSI/SCEIO Conference, University of Victoria, British Columbia, 1980.

[Lesperance 80b] Lesperance, Yves. Exception Handling in PSN. M.Sc. Thesis, Department of Computer Science, University of Toronto, 1980.

[Levesque 77] Levesque, Hector J. A Procedural Approach to Semantic Networks. Tech. Report No. 105, Dept. of Computer Science, University of Toronto, 1977.

[Levesque and Mylopoulos 79] Levesque, Hector J. and John Mylopoulos. "A Procedural Semantics for Semantic Networks", in Associative Networks: Representation and use of knowledge by computers, ed. Nicholas V. Findler. New York: Associated Press, 1979.

[Levesque et al 76] Levesque, Hector J., John Mylopoulos, Gordon I. McCalla, Lucio Melli, and John K. Tsotsos. "A Formalism for Modelling". Proceedings 1st CSCSI/SCEIO Conference, University of British Columbia, Vancouver, B. C., 1976.

[Minsky 75] Minsky, Marvin. "A Framework for Representing Knowledge", in The Psychology of Computer Vision, ed. Patrick H. Winston. New York: McGraw-Hill, 1975.

[Roberts and Goldstein 77] Roberts, B. P. and Ira P. Goldstein. "The FRL Primer". AI Lab Memo 408, Massachusetts Institute of Technology, 1977.

[Schneider 78a] Schneider, Peter F. Organization of Knowledge in a Procedural Semantic Network Formalism. Tech. Report No. 115, Dept. of Computer Science, University of Toronto, 1978.

[Schneider 78b] Schneider, Peter F. "Organization of Knowledge for a Procedural Semantic Network Formalism". Proceedings 2nd CSCSI/SCEIO Conference, University of Toronto, Ontario, 1978.

[Schneider 80] Schneider, Peter F. "A Formal Definition of Contexts in the PSN Formalism". AI Memo 79-5, AI Group, Dept. of Computer Science, University of Toronto, 1980.

[Shibahara et al 82] Shibahara, T., J. Mylopoulos, H. D. Covvey, and J. Tsotsos. "CAA: A knowledge based system with causal knowledge to diagnose rhythm disorders in the heart". Proceedings 4th CSCSI/SCEIO Conference, University of Saskatoon, Saskatchewan, 1982.

[Tsotsos 80] Tsotsos, John K. A Framework for Visual Motion Understanding. Tech. Rep. CSRG-114, Computer Systems Research Group, University of Toronto, 1978.

A COMPUTATIONAL APPROACH TO FUZZY QUANTIFIERS IN NATURAL LANGUAGES[1]

Lotfi A. Zadeh

Division of Computer Science
University of California, Berkeley, CA 94720

ABSTRACT

The generic term fuzzy quantifier is employed in this paper to denote the collection of quantifiers in natural languages whose representative elements are: several, most, much, many, not many, very many, not very many, few, quite a few, large number, small number, close to five, approximately ten, etc. In our approach, such quantifiers are treated as fuzzy numbers which may be manipulated through the use of fuzzy arithmetic and, more generally, fuzzy logic.

A concept which plays an essential role in the treatment of fuzzy quantifiers is that of the cardinality of fuzzy sets. Through the use of this concept, the meaning of a proposition containing one or more fuzzy quantifiers may be represented as a system of elastic constraints whose domain is a collection of fuzzy relations in a relational database. This representation, then, provides a basis for inference from premises which contain fuzzy quantifiers. For example, from the propositions "most X's are A's" and "most A's are B's," it follows that "most X's are B's" where ^{2}most is the fuzzy product of the fuzzy proportion most with itself. The method in question may be viewed as a constituent of test-score semantics-- a meaning-representation system for natural-languages in which the meaning of a semantic entity is represented as a procedure which tests, scores and aggregates the elastic constraints which are induced by the entity in question.

1. INTRODUCTION

During the past decade, the work of Montague and others (Montague (1974), Partee (1976), Dowty (1981)) has contributed greatly to our understanding of the proper treatment of the quantifiers all, some and any when they occur singly or in combination in a proposition in a natural language.

Recently, Barwise and Cooper and others (Barwise and Cooper (1981), Peterson (1980)) have described methods of dealing with so-called generalized quantifiers exemplified by most, many, etc. In a different approach which we have described in a series of papers starting in 1975 (Zadeh (1975, 1977, 1978, 1981)), the quantifiers in questions -- as well as other quantifiers with imprecise meaning such as few, several, not very many, etc. are treated as fuzzy numbers and hence are referred to as fuzzy quantifiers. As an illustration, a fuzzy quantifier such as most in the proposition "Most tall men are fat" is interpreted as a fuzzily defined proportion of the fuzzy set of fat men in the fuzzy set of tall men. Then, the concept of the cardinality of fuzzy sets is employed to compute the proportion in question and find the degree to which it is compatible with the meaning of most.

A convenient framework for the treatment of fuzzy quantifiers as fuzzy numbers is provided by a recently developed meaning-representation system for natural languages termed test-score semantics (Zadeh (1981)).

Test-score semantics represents a break with the traditional approaches to semantics in that it is based on the premise that almost everything that relates to natural languages is a matter of degree. The acceptance of this premise necessitates an abandonment of bivalent logical systems as a basis for the analysis of natural languages and suggests the adoption of fuzzy logic (Zadeh (1975, 1979), Bellman and Zadeh (1979)) as the basic conceptual framework for the representation of meaning, knowledge and strength of belief.

Viewed from the perspective of test-score semantics, a semantic entity such as a proposition, predicate, predicate-modifier, quantifier, qualifier, command, question, etc., may be regarded as a system of elastic constraints whose domain is a collection of fuzzy relations in a database -- a database which describes a state of affairs (Carnap (1952)) or a possible world (Lambert and van Fraassen (1970)) or, more generally, a set of objects or derived objects in a universe of discourse. The meaning of a semantic entity, then, is represented as a test which applied to the database yields a collection of partial test scores. Upon aggregation, these test scores lead to an overall vector test score, τ, whose components are numbers in the unit interval, with τ serving as a measure of the compatibility of the semantic entity with the database. In this respect, test-score semantics subsumes both truth-conditional and possible-world semantics as limiting cases in which the partial and overall test scores are restricted to {pass, fail} or, equivalently, {true, false} or {1,0}.

In more specific terms, the process of meaning re-

[1]Research supported in part by the NSF Grants MCS79--6543 and IST-801896

presentation in test-score semantics involves three distinct phases. In Phase 1, an explanatory database frame or EDF, for short, is constructed. EDF consists of a collection of relational frames, i.e., names of relations, names of attributes and attribute domains whose meaning is assumed to be known. In consequence of this assumption, the choice of EDF is not unique and is strongly influenced by the knowledge profile of the addressee of the representation process as well as by the desideratum of explanatory effectiveness. For example, in the case of the proposition p $\triangleq$ Over the past few years Nick earned far more than most of his close friends, the EDF might consist of the following relations: INCOME [Name; Amount; Year] which lists the income of each individual identified by his/her name as a function of the variable Year; FRIEND [Name; μ], where μ is the degree to which Name is a friend of Nick; FEW [Number; μ], where μ is the degree to which Number is compatible with the fuzzy quantifier few; MOST [Proportion; μ], in which μ is the degree to which Proportion is compatible with the fuzzy quantifier most; and FAR.MORE [Income1; Income2; μ], where μ is the degree to which Income1 fits the fuzzy predicate far more in relation to Income2. Each of these relations is interpreted as an elastic constraint on the variables which are associated with it.

In Phase 2, a test procedure is constructed which acts on relations in the explanatory database and yields the test scores which represent the degrees to which the elastic constraints induced by the constituents of the semantic entity are satisfied. For example, in the case of p, the test procedure would yield the test scores for the constraints induced by the relations FRIEND, FEW, MOST and FAR.MORE.

In Phase 3, the partial test scores are aggregated into an overall test score, τ, which, in general, is a vector which serves as a measure of the compatibility of the semantic entity with an instantiation of EDF. As was stated earlier, the components of this vector are numbers in the unit interval or, more generally, possibility/probability distributions over this interval. In particular, in the case of a proposition, p, for which the overall test score is a scalar, τ may be interpreted -- in the spirit of truth-conditional semantics -- as the degree of truth of the proposition with respect to the expalantory database ED (i.e., an instantiation of EDF). Equivalently, τ may be interpreted as the possibility of ED given p, in which case we may say that p induces a possibility distribution. More concretely, we shall say that p translates into a possibility assignment equation (Zadeh (1978)):

$$p \rightarrow \Pi_{(X_1,\cdots,X_n)} = F$$

where F is a fuzzy subset of a universe of discourse U, $X_1,\cdots,X_n$ are variables which are explicit or implicit in p, and $\Pi_{(X_1,\cdots,X_n)}$ is their joint possibility distribution. For example, in the case of the proposition p $\triangleq$ Danielle is tall, we have

$$\text{Danielle is tall} \rightarrow \Pi_{\text{Height(Danielle)}} = \text{TALL} \qquad (1.1)$$

where TALL is a fuzzy subset of the real line, Height(Danielle) is a variable which is implicit in p, and $\Pi_{\text{Height(Danielle)}}$ is the possibility distribution of the variable Height(Danielle). Equation (1.1) implies that

$$\text{Poss}\{\text{Height(Danielle)}=u\} = \mu_{\text{TALL}}(u)$$

where u is a specified value of the variable Height(Danielle), $\mu_{\text{TALL}}(u)$ is the grade of membership of u in the fuzzy set TALL, and Poss(X=u) should be read as "the possibility that X is u." In effect, (1.1) signifies that the proposition "Danielle is tall," may be interpreted as an elastic constraint on the variable Height(Danielle), with the elasticity of the constraint characterized by the unary relation TALL[2] which is defined as a fuzzy subset of the real line.

The same basic idea may be applied to propositions containing one or more fuzzy quantifiers. As a simple illustration, let us consider the proposition

p$\triangleq$ Mary has several credit cards

in which several is regarded as a fuzzy quantifier which induces an elastic constraint on the number of credit cards possessed by Mary. In this case, X may be taken to be the count of Mary's cards, and the possibility assignment equation becomes

$$\text{Mary has several credit cards} \rightarrow \Pi_{\text{Count(Cards(Mary))}} = \text{SEVERAL}, \qquad (1.2)$$

in which SEVERAL plays the role of a specified fuzzy subset of the integers 1, 2,..., 10. Thus, if the integer 4, say, is assumed to be compatible with the meaning of several to the degree 0.8, then (1.2) implies that, given p and the definition of several, the possibility that Mary has four credit cards is expressed by

$$\text{Poss}(\text{Count(Cards(Mary))}=4) = 0.8.$$

In the above example, the class of Mary's credit cards is a nonfuzzy set and hence there is no problem in counting their number. By contrast, in the proposition

p$\triangleq$ Mary has several close friends

the class of close friends is a fuzzy set and thus we must first resolve the question of how to count the number of elements in a fuzzy sets or, equivalently, how to determine its cardinality. This issue is addressed in the following section.

2. CARDINALITY OF FUZZY SETS

For simplicity, we shall restrict our attention to finite universes of discourse, in which case a fuzzy subset of $U=\{u_1,\cdots,u_n\}$ may be expressed symbolically as

$$F=\mu_1/u_1+\cdots+\mu_n/u_n$$

in which the term μ_i/u_i, $i=1,\cdots,n$, signifies that μ_i is the grade of membership of u_i in F, and

[2]Generally, we follow the practice of writing the names of fuzzy subsets and fuzzy relations in uppercase symbols.

the plus sign represents the union.[3]

A simple way of extending the concept of cardinality to fuzzy sets is to form the sigma-count (Zadeh(1978, 1981)), which is the arithmetic sum of the grades of membership in F. Thus

$$\Sigma\text{Count}(F) \triangleq \Sigma_i \mu_i$$

with the understanding that the sum may be rounded, if need be, to the nearest integer. Furthermore, one may stipulate that the terms whose grade of membership falls below a specified threshold be excluded from the summation. The purpose of such an exclusion is to avoid a situation in which a large number of terms with low grades of membership become count-equivalent to a small number of terms with high membership.

As a simple illustration of the concept of sigma-count, assume that the fuzzy set of close friends of Mary is expressed as

$$F=0.8/\text{Enrique}+1/\text{Ramon}+0.7/\text{Elie}+0.9/\text{Sergei}+0.8/\text{Ron}$$

In this case,

$$\Sigma\text{Count}(F) = 0.8+1+0.7+0.9+0.8 = 4.2$$

Another and perhaps more natural approach is to allow the cardinality of a fuzzy set to be a fuzzy number (Zadeh (1979)). In this case, the point of departure is a stratified representation of F in terms of its level sets, i.e.,

$$F = \Sigma_\alpha \alpha F_\alpha$$

in which the α-level-sets F_α are nonfuzzy sets defined by

$$F_\alpha \triangleq \{u \mid \mu_F(u) \geq \alpha\} \quad , \quad 0 < \alpha \leq 1 \ .$$

In terms of this representation, there are three fuzzy counts that may be associated with F. First, the FGCount is defined as the fuzzy integer

$$\text{FGCount}(F) = \Sigma_\alpha \alpha/\text{Count}(F_\alpha)$$

Second, the FLCount is defined as

$$\text{FLCount}(F) = (\text{FGCount}(F))' \ominus 1$$

where ' denotes the complement, and $\ominus 1$ means that 1 is subtracted from the fuzzy number FGCount (F). And finally, the FECount (F) is defined as the intersection of FGCount(F) and FLCount(F), i.e.,

$$\text{FECount}(F) = \text{FGCount}(F) \cap \text{FLCount}(F)$$

Equivalently, we may define the counts in question via their membership function, i.e.,

$$\mu_{\text{FGCount}(F)}(i) \triangleq \sup_\alpha \{\alpha \mid \text{Count}(F_\alpha) \geq i\}, \ i=0,1,\cdots,n$$

$$\mu_{\text{FLCount}(F)}(i) \triangleq \sup_\alpha \{\alpha \mid \text{Count}(F'_{1-\alpha}) \geq n-i\}$$

$$\mu_{\text{FECount}(F)}(i) \triangleq \mu_{\text{FGCount}(F)}(i) \wedge \mu_{\text{FLCount}(F)}(i)$$

[3]For the most part we shall rely on the context to disambiquate the meaning of +.

where $\wedge$ stands for min in infix position

For our purposes, it will be sufficient to deal with FGCount(F). For this count, $\mu_{\text{FGCount}(F)}(i)$ may be interpreted as the truth-value of the proposition "The number of elements in F is greater than or equal to i." Another useful observation is that FGCount(F) may readily be obtained from F by first sorting F in the order of decreasing grades of membership and then replacing u_1 with 1. For example, if

$$F=0.6/u_1+0.9/u_2+1/u_3+0.7/u_4+0.3/u_5$$

then

$$\text{FGCount}(F)=1/1+0.9/2+0.7/3+0.6/4+0.3/5$$

As was stated earlier, the concept of cardinality of fuzzy sets plays an essential role in testscore semantics in representing the meaning of semantic entities containing fuzzy quantifiers. In the following section, we shall consider a few examples which serve to illustrate the meaning-representation process in question.

3. MEANING REPRESENTATION AND INFERENCE

Consider the semantic entity

SE $\triangleq$ several balls most of which are large

For this semantic entity, we shall assume that EDF comprises the following relations:

$$\begin{aligned}\text{EDF} \triangleq\ & \text{BALL[Identifier Size]} + \\ & \text{LARGE[Size; } \mu\text{]} + \\ & \text{SEVERAL[Number; } \mu\text{]} + \\ & \text{MOST[Proportion; } \mu\text{]}\end{aligned}$$

In this EDF, the first relation has n rows and is a list of the identifiers of balls and their respective sizes; in LARGE, μ is the degree to which a ball of size Size is large; in SEVERAL, μ is the degree to which Number fits the description several; and in MOST, μ is the degree to which Proportion fits the description most.

The test which yields the compatibility of SE with ED and thus defines the meaning of SE depends on the definition of fuzzy set cardinality. In particular, using the sigma-count, the test procedure becomes:

1. Test the constraint induced by SEVERAL:

$$\tau_1 = \mu_{\text{SEVERAL[Number=n]}}$$

which means that the value of Number is set to n and the value of μ is read, yielding the test score τ_1 for the constraint in question.

2. Find the size of each ball in BALL:

$$\text{Size}_i = \text{Size}_{\text{BALL[Identifier=Identifier}_i\text{]}}, \ i=1,\cdots,n.$$

3. Test the constraint induced by LARGE for each ball in BALL:

$$\mu_{LB}(i) = \mu_{\text{LARGE[Size=Size}_i\text{]}}$$

4. Find the sigma-count of large balls in BALL:

$$\Sigma\text{Count}(LB) = \Sigma_i \mu_{LB}(i)$$

5. Find the proportion of large balls in BALL:

$$PLB = \frac{1}{n} \Sigma_i \mu_{LB}(i)$$

6. Test the constraint induced by MOST:

$$\tau_2 = {}_{\mu}\text{MOST}[\text{Proportion}=PLB]$$

7. Aggregate the partial test scores:

$$\tau = \tau_1 \wedge \tau_2$$

where τ is the overall test score. The use of the min operator to aggregate τ_1 and τ_2 implies that we interpret the implicit conjunction in SE as the cartesian product of the conjuncts.

The use of fuzzy cardinality affects the way in which τ_2 is computed. Specifically, the employment of FGCount leads to:

$$\tau_2 = \sup_i (\text{FGCount}(LB) \cap n \times \text{MOST})$$

which expressed in terms of the membership functions of FGCount(LB) and MOST may be written as

$$\tau_2 = \sup_i (\mu_{\text{FGCount}(LB)}(i) \wedge \mu_{\text{MOST}}(\tfrac{i}{n}))$$

The rest of the test procedure is unchanged.

As an additional example, consider the proposition

p≜ over the past few years Nick earned far more than most of his close friends

In this case, we shall assume that EDF consists of the following relations:

EDF≜ INCOME[Name; Amount; Year]+
FRIEND[Name; μ]+
FEW[Number; μ]+
FAR.MORE[Income1; Income2; μ]+
MOST[Proportion; μ]

Using the sigma-count, the test procedure may be described as follows:

1. Find Nick's income in Year_i, i=1,2,···, counting backward from present:

$$IN_i \triangleq {}_{\text{Amount}}\text{INCOME}[\text{Name}=\text{Nick}; \text{Year}=\text{Year}_i]$$

2. Test the constraint induced by FEW:

$$\mu_i \triangleq {}_{\mu}\text{FEW}[\text{Year}=\text{Year}_i]$$

3. Compute Nick's total income during the past few years:

$$TIN = \Sigma_i \mu_i IN_i$$

in which the μ_i play the role of weighting coefficients.

4. Compute the total income of each Name_j (other than Nick) during the past several years:

$$\text{TIName}_j = \Sigma_i \mu_i \text{IName}_{ji}$$

where IName_{ji} is the income of Name_j in Year_i

5. Find the fuzzy set of individuals in relation to whom Nick earned far more. The grade of membership of Name_j in this set is given by

$$\mu_{FM}(\text{Name}_j) = {}_{\mu}\text{FAR.MORE}[\text{Income1}=TIN; \text{Income2}=\text{TIName}_j]$$

6. Find the fuzzy set of close friends of Nick by intensifying (Zadeh (1978)) the relation FRIEND:

$$CF = \text{FRIEND}^2$$

which implies that

$$\mu_{CF}(\text{Name}_j) = ({}_{\mu}\text{FRIEND}[\text{Name}=\text{Name}_j])^2$$

7. Using the sigma-count, count the number of close friends of Nick:

$$\Sigma\text{Count}(CF) = \Sigma_j \mu^2_{\text{FRIEND}}(\text{Name}_j)$$

8. Find the intersection of FM with CF. The grade of membership of Name_j in the intersection is given by

$$\mu_{FM \cap CF}(\text{Name}_j) = \mu_{FM}(\text{Name}_j) \wedge \mu_{CF}(\text{Name}_j)$$

9. Compute the sigma-count of FM∩CF:

$$\Sigma\text{Count}(FM \cap CF) = \Sigma_j \mu_{FM}(\text{Name}_j) \wedge \mu_{CF}(\text{Name}_j)$$

10. Compute the proportion of individuals in FM∩CF who are in CF:

$$P \triangleq \frac{\Sigma\text{Count}(FM \cap CF)}{\Sigma\text{Count}(CF)}$$

11. Test the constraint induced by MOST:

$$\tau = {}_{\mu}\text{MOST}[\text{Proportion}=P]$$

which expresses the overall test score and thus represents the desired compatibility of p with the explanatory database.

The remaining examples illustrate how one can infer from propositions containing fuzzy quantifiers.

Example

p_1≜ Dana is about 22
p ≜ Tandy is a few years older than Dana

How old is Tandy?

Interpreting the fuzzy quantifier about 22 as a fuzzy number and employing fuzzy arithmetic (Dubois and Prade (1980)), Tandy's age may be expressed as the fuzzy sum

$$\text{Age (Tandy)} = \text{ABOUT } 22 \oplus \text{FEW}$$

The possibility distribution induced by the fuzzy number Age (Tandy) is given by

$$\mu_{\text{Age(Tandy)}}(v) = \sup_u(\mu_{\text{ABOUT } 22}(u) \wedge \mu_{\text{FEW}}(v-u))$$

Example

p ≜ most Frenchmen are not very tall

How many Frenchmen are tall?

As shown in Zadeh (1978), p is semantically equivalent to the proposition

≜ ant(most) Frenchmen are very tall

where ant(most) denotes the antonym of most, i.e.,

$$\mu_{ANT(MOST)}(u) = \mu_{MOST}(1-u), \qquad u \in [0,1]$$

Furthermore, it can readily be shown that, in general, a proposition of the form

QX's are very F

where Q is a fuzzy quantifier and F is a fuzzy set, entails

$$r \triangleq Q^{\frac{1}{2}}\ X\text{'s are } F$$

where $Q^{\frac{1}{2}}$ is the square root of the fuzzy number Q. Consequently, the answer to the question may be expressed as

$$(ant(most))^{\frac{1}{2}}\ \text{Frenchmen are tall:}$$

where

$$\mu_{(ANT(MOST))^{\frac{1}{2}}}(u) = \mu_{MOST}((1-u)), \qquad u \in [0,1].$$

The main point that the above examples are intended to make is that the interpretation of fuzzy quantifiers as fuzzy numbers provides a systematic basis for both representing the meaning of - and inferring from- propositions containing fuzzy quantifiers. In the case of inference, the answer to a question is, in general, a possibility distribution which may be viewed as an elastic restriction on the possible values of the variable in question.

REFERENCES AND RELATED PUBLICATIONS

1. Barwise, J. and Cooper, R., Generalized Quantifiers and Natural Language," Linguistics and Philosophy 4 (1981) 159-219.
2. Bellman, R. E. and Zadeh, L. A., Local and Fuzzy Logics, in: Modern Uses of Multiple-Valued Logic, Epstein, G., (ed.). Dordrecht: (D. Reidel 103-165, 1977).
3. Carnap, R., Meaning and Necessity (University of Chicago Press, 1952).
4. Cresswell, M. J., Logic and Languages (London: Methuen, 1973).
5. DeLuca, A. and Termini, S., A definition of a non-probabilistic entropy in the setting of fuzzy sets theory, Information and Control 20 (1972) 301-312.
6. Dowty, D. R., et al, Introduction to Montague Semantics, Dordrecht: (Reidel, 1981).
7. Dubois, D. and Prade, H., Fuzzy Sets and Systems: Theory and Applications, New York: Academic Press, 1980.
8. Gaines, B. R., Logical foundations for database systems, Int. J. Man-Machine Studies II (1979) 481-500.
9. Hobbs, Making computational sense of Montague's intensional logic, Artificial Intelligence 9 (1978) 287-306.
10. Lakoff, G., Hedges: a study in meaning criteria and the logic of fuzzy concepts, J. Phil. Logic 2 (1973) 458-508. Also in: Contemporary Research in Philosophical Logic and Linguistic Semantics (D. Jockney, W. Harper and B. Freed, eds.). Dordrecht: (D. Riedel, 221-271, 1973)
11. Lambert, K. and van Fraassen, B. C., Meaning Relations, Possible Objects and Possible Worlds, Philosophical Problems in Logic (1970) 1-19.
12. McCawley, J. D., Everything that Linguists have Always Wanted to Know about Logic (University of Chicago Press, 1981).
13. Montague, R., Formal Philosophy, in:Selected Papers, Thomason, R., (ed.). New Haven: (Yale University Press, 1974).
14. Mizumoto, M. and Tanaka, K., Some properties of fuzzy numbers, in: Advances in Fuzzy Set Theory and Applications, Gupta, M. M., Ragade, R. K., and Yager, R. R., eds.). Amsterdam: North Holland, 153-164, 1979.
15. Partee, B., Montague Grammar (New York: Academic Press, 1976).
16. Peterson, P., On the logic of few, many and most, Notre Dame J. of Formal Logic 20 (1979) 155-179.
17. Schubert, L. K., Goebel, R. G. and Cercone, N., The structure and organization of a semantic net for comprehension and inference, in: Associative Networks, Findler, N. V.(ed.) New York: Academic Press, 122-178, 1979.
18. Zadeh, L. A., Fuzzy logic and approximate reasoning, Synthese 30 (1975) 407-428.
19. Zadeh, L. A., A theory of approximate reasoning, in: Electronics Research Laboratory Memorandum M77/58, University of California, Berkeley, 1977. Also in: Machine Intelligence 9, Hayes, J. E., Michie, D. and Kulich, L. I., (eds.). New York: (Wiley, 149-194,1979)
20. Zadeh, L. A., PRUF--a meaning representation language for natural languages, Int. J. Man-Machine Studies 10 (1978) 395-460.
21. Zadeh, L. A., Test-score semantics for natural languages and meaning-representation via PRUF, Tech. Note 247, AI Center, SRI International, Menlo Park, CA., 1981.
22. Zimmer, A., Some experiments concerning the fuzzy meaning of logical quantifiers, in: General Surveys of Systems Methodology, Troncoli, L. (ed.). Louisville: Society for General Systems Research 435-441, 1982.

GOAL SELECTION STRATEGIES IN HORN CLAUSE PROGRAMMING

E.W. Elcock

The University of Western Ontario

It is arguable that knowledge representation and use should be founded on a complete system. For example, if the knowledge representation language is to be first-order logic, then we would like to express the knowledge K under the assumption that we have available a sequenthood procedure which is complete in the sense that any true sequent K => G is demonstrably true by the procedure. For example, our knowledge system might be based on finite clausal sequents for which there is indeed a procedure using resolution which has the completeness property (Robinson, 1979).

For resolution systems it is well-known that selection strategies play a vital role in determining the pragmatics of such systems and design of strategies has been an ongoing research activity.

Over the last decade an incomplete system called Prolog has been elaborated and has become widely used. Prolog has intriguing analogies with Absys - an assertative programming system developed in 1968 (Foster and Elcock, 1969). This note attempts to illustrate some issues of incompleteness by comparing some aspects of the two systems.

Prolog is a well documented system. There are excellent texts (Kowalski, 1979; Clocksin and Mellish, 1981), together with numerous short overviews. For this reason we shall simply note here that a Prolog program is a pair [A,G] where A is a sequence of Horn clauses and G a conjunction of literals. The pair [A,G] has the model theoretic reading that A => G is a true sequent (G logically follows from A). It has a procedural reading which regards G as an executable statement which is interpreted as a set of procedure calls to procedures declared in A. The execution of the procedure calls essentially parallels a linear input resolution strategy. The model theoretic and procedural semantics are potentially equivalent in a precise sense (Van Emden and Kowalski, 1976). However, for pragmatic reasons, Prolog uses a procedure call and evaluation strategy which parallels the LUSH restriction of the linear input strategy. Essentially, the current executable statement is regarded, not as a set, but as a sequence of procedure calls, and A is regarded as a sequence of procedure declarations. Prolog attempts to execute the first of the sequence of calls using the first matching declaration in the sequence of procedure declarations - the parallel of the depth first LUSH resolution strategy. Successful matching leads to replacement of the executed call by the body of the relevant declaration, and the resumption of the execution cycle with an augmented binding environment determined by the matching (for details see Kowalski, 1979).

This execution strategy makes Prolog incomplete in the sense that a true sequent A => G is not necessarily a terminating Prolog program [A,G]. A simple illustrative example is the program [A,G] where A is the sequence of clauses

1. mem(X,[Y|L]) if mem(X,L).

2. mem((X,[X|L].

specifying list membership, and G is

mem(a,[a|L]).

[X|L] is simply a special notation (used in the Edinburgh Prolog) for a term whose intended interpretation is a list with first member X and remainder list L.

The sequent A => G is true but the Prolog evaluator persistently uses clause 1 of A to generate the sequence of execution statements

mem(a,[a|L])

mem(a,L)

mem(a,L1) where L is [X|L1]

mem(a,L2) where L1 is [Y|L2]

etc. etc.

Unlike Prolog, Absys is not a well documented system and only informal accounts are available (Foster and Elcock, 1969; Elcock, 1971). Absys (standing for Aberdeen System) was an experimental working on-line incremental compiler for assertions, developed by the Computer Research Group at the University of Aberdeen and essentially completed in 1968.

A written program in Absys consists of a conjunction of assertions about objects and relations holding over them. The system acts to construct objects satisfying the conjunction of assertions. The written program places no explicit constraints on the order in which particular operations are performed. In addition,

the effect of processing an assertion, as in Prolog, depends upon the binding context in which the assertion is processed. Thus, in Absys, as in Prolog,

```
L = [X|M]
```

simply asserts that L is a list whose head is X and whose tail is L. Whether the assertion acts to construct L, or to select X and M, or simply check that L,X and M satisfy the asserted relation, depends solely on the data or binding environment at the time that the assertion is processed.

In Absys alternatives could be asserted by an explicit disjunction <<a1 or a2>> where a1 and a2 are conjunctions of assertions. The (implicit) and and or distribute in the usual way so that, for example,

```
a1 << a2 or a3 >> a4
```

is equivalent to

```
<< a1 a2 a4 >>  or  << a1 a3 a4 >> .
```

The system attempts to construct data to satisfy each conjunction of assertions, each conjunction notionally constituting a separate (parallel) computation branch of the total program. In practice, of course, the non-determinism was handled by appropriate differential record keeping and backtracking in a similar spirit to Prolog implementations. The distribution of and and or connectives was handled in a way which attempted to minimize duplication of processing. A particular computational branch terminates when unsatisfiability is detected.

A lambda construction allowed the expression of functions other than the primitives of the system - the analogy of the procedure declarations of Prolog. Thus list membership might be specified in Absys by:

```
mem = lambda m,s key  s
        << s = [p|s1] and
           << m = p or mem(m,s1) >> >>
```

The assertion

```
mem(x,[1,2,3])
```

is equivalent to asserting

```
<< x=1 or x=2 or x=3 >> .
```

If in addition we were now to assert

```
mem(x,[2,4,6])
```

equivalent to

```
<< x=2 or x=4 or x=6 >> ,
```

then distribution would lead to nine computational branches of which only one, that associated with x=2 would be satisfiable and hence remain active.

The "key" statement in the declaration of "mem" (elaborated later on) prevents the kind of non-terminating behaviour exhibited in the (admittedly deliberately contrived) Prolog specification of "mem". The "key" statement in effect says "don't elaborate this call of mem unless the actual parameter which is to be bound to the indicated formal parameter "s" of mem has a value". This prevents a possible attempt to elaborate the recursive call of mem before the conjoined assertion s = [p|s1] has been successfully elaborated, and hence prevents the possibility of non-terminating elaboration of mem.

Anticipating Prolog, Absys had a primitive aggregation operator set: it takes as parameters a prototype set element and an assertion and produces the set of prototype elements satisfying the assertion. The assertion is typically a disjunction. The set aggregator initiates the "parallel" computations and then extracts the datum corresponding to the prototype from those computations which do not terminate because of unsatisfiability (c.f. Prolog's extraction of those elements for which the assertion is established to be a logical consequence of the "A" sequence of clauses).

Anticipating Prolog and Planner, Absys negation acts like a degenerate or in that it initiates an independent computational branch, but one in which the criteria for termination are reversed in that not (<<a>>) is satisfiable iff <<a>> is unsatisfiable (c.f. Prolog's "not provable").

Although neither the negation nor the aggregation operator are central to the theme of this note, they are mentioned in passing in the spirit of historical footnotes of potential interest to a new generation of researchers in artificial intelligence! The main thrust of this thumbnail sketch of Absys, which draws heavily on Foster and Elcock (1969), is to bring attention to the fact that Absys, like Prolog, has a declarative reading (semantics) which asserts relations holding over objects. Also like Prolog, it has a uniform evaluation mechanism (inducing a procedural semantics) which attempts to instantiate variables by constants in such a way that the assertions are demonstrably satisfied. Unlike Prolog, however, Absys, although possessing a clean applicative semantics, lacks the generality of Prolog's powerful theoretical underpinnings supplied by the model theory of first-order logic and the practical power of unification as the basis for the procedure calling mechanism for the system viewed as a programming language.

We have drawn attention to the fact that the Prolog evaluation mechanism is incomplete.

What about the Absys evaluation mechanism? It too is incomplete, but for quite different reasons and in an interestingly different way.

The Absys evaluation mechanism is explained in detail in Elcock et.al (1972). Absys maintains a list of relations still to be elaborated. Suppose we activate the first such relation f(x,y,z) , say, on this list. It is expected that the functor f , whether primitive or user defined, has been defined in such a way that the header for f specifies, by means of its "key" statement, a constraint on its argument set which indicates whether or not it is "worthwhile" elaborating the body of f (e.g. "plus(x,y,z)" is only worthwhile elaborating if two of its arguments already have values in the domain of "plus"). If the relation is not deemed worth elaborating, then it is "associated" with each of its arguments which are currently uninstantiated variables. The evaluator now continues processing the list of relations awaiting elaboration. Suppose, on the other hand, that the relation f(x,y,z) is worth processing in the sense that enough is known about some of the arguments to allow others to be inferred through elaboration of the body of the function. The body is now elaborated and the binding environment necessarily augmented. The process of changing a variable binding in the binding environment automatically returns any relations associated with that variable to the list of relations still to be elaborated.

Let us illustrate this with a very simple example involving only system functions. Suppose we have the Absys program

u+v = 16 and w*u = v and u+10 = 12

and suppose the list of relations is initially processed in this order. The relation u+v = 16 is examined and, for the reasons mentioned, associated with u and v . The relation w*u = v is now examined and associated with w,u and v . We now have the association lists (u: u+v = 16; w*u = v) ; (v: u+v = 16; w*u = v) and (w: w*u = v). The relation u+10 = 12 is now examined, elaborated and the binding environment augmented so that u is bound to 2 (12-10). As a result of this binding to u , the list of relations (2+v = 16; w*2 = v) associated with u is appended to the list of relations to be elaborated. The relation 2+v = 16 is now examined and elaborated and the binding environment augmented so that v is bound to 14 , and the list associated with v appended to the list of relations to be elaborated. This list is now (2+14 = 16; w*2 = 14; w*2 = 14). The first relation is examined, elaborated and found satisfied. The second is examined, elaborated and the binding environment augmented so that w is bound to 7 , and the list associated with w appended to the list of relations still to be processed. This list is now (7*2 = 14; 7*2 = 14). Both these remaining relations are examined, elaborated and found satisfied. The list of relations to be elaborated is now empty and the (examinable) state of the binding environment reflects what the system has been able to infer from the original conjunction of assertions.

Let's now contrast this with a Prolog evaluation. Suppose we ask Prolog to establish that its "system axioms" imply that there exists a u,v, and w such that

u+v = 16 and w*u = v and u+10 = 12 .

If the goal conjunction is in this order, then all Prolog implementions of which I am aware would fail at the first relation. This is because top-down-left-right relation selection-elaboration strategy insists that Prolog determine a successful match for the selected relation and elaborate it or else fail. Now Prolog, like Absys, sensibly says that it is not going to have a system specification of "+" which will allow a matching of u+v = 12 involving an infinite (or at least potentially very very large) set of pairs u,v satisfying the relation: the chance of doing any useful arithmetic this way are slim. The Edinburgh Prolog, excessively cautious in the true Scots tradition, would insist that all of u,v, and w are already instantiated to integers. IC Prolog from swinging London, like Absys, is happy if two of the three variables are instantiated by integers at the time of elaborating the call. However, IC Prolog still could not cope with the above ordering because of the left-right rule, although it could cope with the logically equivalent conjunction

u+10 = 12 and u*v = 16 and w*u = v.

Indeed, the action of the Absys evaluator could be viewed as dynamically rearranging the order of elaboration of the relations under the influence of the changing binding environment. We are now at the heart of the matter.

Certainly the arithmetic example, of itself, is not very exciting. However, the illustrated problem is quite general. It is that a knowledge manipulation system is likely to have enough to worry about to generate a specification of a consequent under its declarative reading without having to worry about any potential incompleteness of a concomitant procedural reading.

For example, the arithmetic relations in the example above might have been generated in that order as a result of a particular parse of the word problem: "Two straight rods laid end to end measure sixteen inches in length. The second rod is longer than the first by a factor "k" , and the first rod is the piece that was left after cutting ten inches of a rod one foot long."

Absys would accept the parsed sequence of relations as is. Prolog would need a further stage of processing in which the conjuncts were reordered to meet certain deficiences in the sequent processor.

The example has, of course, been chosen to show Absys to advantage. However, the Absys dynamic data directed elaboration of the conjunction is gained only at a high price of more elaborate runtime processing structures. In any case, a naive dynamic data directed flow of elaboration, although elegant in certain well circumscribed contexts, of itself rapidly runs out of steam. For example, and staying within arithmetic for pedagogic simplicity, much more sophisticated aggregation and solution methods would be necessary to deal with a conjunction of a general set of linear equations in n variables. Even the task of recognizing what aggregations of individual conjuncts might lend themselves to reorganization as a specified "higher" relation (e.g. the relation "simult-eqs(L)" where L is a list of lists of coefficients say) is challenging to say the least.

Indeed such aggregation is a central problem of knowledge deployment. Nevertheless, it is likely that flexible dynamically determined selection strategies for evaluation systems will remain an important feature of good knowledge manipulation systems in whatever formalism. The dynamic methods of Absys take one a little way. The author is currently investigating whether such methods can be extended and embedded in the context of a suitably designed logic programming language.

Summary

A major problem with particular logic programming languages is that a sequent may be true but not established as such by the system simply because, in the interests of certain notions of efficiency, the sequenthood establishing procedure used by the system is incomplete.

It has been argued that a central issue for Prolog (and for other first-order systems) as a vehicle for knowledge representation and use is the dynamic aggregation of and selection of appropriate relations for elaboration. The issue has been illustrated by a comparison of the elaboration strategies of Abset and Prolog. Examination of the work of (certain) members of the Prolog programming community shows that incredibly complex procedural effects can be obtained (often by the rape of any model theoretic semantics the constructs might once have had). Although not a sufficient condition, this phenomenon offers hope that sophisticated behaviour might be obtained by tidier means. Indeed, the meta-logical approaches of Kowalski (Kowalski, 1979) and others are examples of such attempts. It might be that, by a suitable superstructure, one could maintain some of the pragmatic advantages of Prolog (or Prolog like systems) and yet avoid the sequencing difficulties identified in this note.

The content of this note and its wider context is part of work being conducted under Operating Grant Number A9123 from the Natural Science & Engineering Research Council of Canada.

References

Clocksin, W.F. and Mellish, C.S. (1981). Programming in Prolog. Springer-Verlag, Berlin.

Elcock, E.W. (1971). Problem solving compilers. Artificial Intelligence and Heuristic Programming. (Ed. Findler, N.). Edinburgh University Press, Edinburgh. pp. 37-50.

Elcock, E.W., McGregor, J.J. and Murray, A.M. (1972). Data directed control and operating systems. British Computer Journal, Vol. 15, No. 2. pp. 125-129.

Foster, J.M. and Elcock, E.W. (1969). Absys 1: an incremental compiler for assertions; an introduction. Machine Intelligence 4, (eds. Meltzer, B. and Michie, D.). Edinburgh University Press, Edinburgh. pp. 423-429.

Kowalski, R.A. (1979). Logic for Problem Solving. North Holland Elsevier, New York.

Robinson, A. (1979). Logic: Form and Function. Edinburgh University Press, Edinburgh.

Van Emden, M.H. and Kowalski, R.A. (1976). The semantics of predicate logic as a programming language. J.A.C.M., Vol. 23, No. 4. pp. 733-742.

AN EXPERIMENTAL THEOREM PROVER USING FAST UNIFICATION AND VERTICAL PATH GRAPHS

Neil V. Murray

LeMoyne College
Syracuse, NY 13214

Abstract

We describe an experimental Prawitz-based theorem prover. Unnormalized quantifier-free formulas are represented as directed acyclic graphs and we use a fast unification algorithm. An additional data structure called a vertical path graph (vpg) is used to guide the prover toward a proof. Paths in a vpg are related to disjuncts in the DNF of the quantifier-free formulas. We define a well-ordering of certain paths in the vpg, which allows the search for a proof to be organized as a recursive backtrack search.

1. Introduction.

In recent years, the non-clausal approach to theorem proving has drawn a growing amount of attention. Some of these efforts are based on inference and/or splitting and reduction methods [4,5,7,12,18] while others are variations of Prawitz analysis [1,2,6,8,16]. The major advantage of the non-clausal approach is that it avoids the proliferation of literals during conversion to conjunctive (or disjunctive) normal form. Two advantages of Prawitz-analysis are that no new formulas are inferred (although variants of formulas may be) and the original problem is not split into separate parts which are then analyzed and processed locally.

The theorem prover described here retains the advantages of non-clausal form and Prawitz analysis. The approach evolved from a preliminary variation on Prawitz-analysis, which was originally inefficient, but is now improved.

In section 2 we briefly describe the original approach and some experimental results. The removal of some major deficiencies is described in section 3. Features of the improved implementation and additional experimental results are presented in section 4, along with a brief discussion of possible future improvements.

2. An approach to Prawitz-analysis based on fast unification and equivalence-class manipulation.

We now tersely describe the formalism on which our theorem prover is based. Atoms are constructed in the usual way from a vocabulary of predicate and function symbols, and variables. Formulas are quantifier-free and all variables are implicitly universally quantified. We place no restriction on truth-functional structure: atoms are formulas and if B and C are formulas, then so are (B & C), (B v C), ~B, (B => C), and (B <=> C). Input to the theorem prover consists of a set of formulas which is interpreted as the conjunction of its members.

Such a set S of formulas is unsatisfiable iff for some n, the set of n variants of S, Sn, has an instance SnM that is a boolean contradiction. Notice that M induces a partition P on the atom set of Sn. Atoms X and Y are in the same block of P iff XM=YM. P is contradictory iff Sn is false under all assignments in which atoms in the same block of P are assigned the same truth-value. Notice also that any partition Q, of which P is a refinement, is contradictory whenever P is. We say a partition is unifiable if it can in fact be induced by some substitution. P is maximal unifiable if it is the refinement of no unifiable partition. Thus Sn has a contradictory instance iff there is a maximal unifiable partition P

of the atom set of Sn which is contradictory.

The partitions of an atom set form a lattice in a natural way. Top is the trivial partition having only one block, and bottom is the other trivial partition in which each block is a singleton. The partial order is containment. To find maximal unifiable partitions, we may begin at top and follow each branch down until we discover the first unifiable partition. Or we may begin at bottom and climb up each branch, until we discover the first partition from which no branch leads to a unifiable partition.

Now given S, we may investigate the maximal unifiable partitions of the atom sets of S1, S2, --- until we discover the first n for which Sn has a unifiable contradictory partition. This approach was directly implemented in the original design of our theorem-prover. We briefly describe the original implementation, noting that some important data structures and techniques mentioned remain an integral part of the improved version.

The set of formulas to be refuted is represented as a set of acyclic graphs (dags), in which different occurrences of the same expression are represented by different paths to the same node. Unifications are performed rapidly through fast unification techniques which make use of the same equivalence class operations (UNION and FIND) as would be needed to represent partitions efficiently [18].

With this representation, climbing up a branch in the lattice structure would involve the merging of two blocks in the current partition. This is done by setting a pointer from the atom representing one block to the atom representing the other, which is also the first step in rapidly testing the resultant partition for unifiability.

However, the partition constructed may be non-unifiable or non-contradictory and thus force backtracking. So the work done by the unification algorithm in the attempt to construct a partition must be quickly reversible. Therefore the sequence of UNION and FIND operations caused by a unification must be remembered and occasionally undone. This would create prohibitive overhead in the best-known implementation of UNION and FIND. To reduce this overhead we do not allow path-compression in FIND, and we ignore the weighted union rule. Omitting path compression increases running time from almost-linear to O(d log n), where d and n are the number of directed arcs and nodes in the dags being unified. The weighted union rule can be used without incurring unacceptable overhead (if UNIONs are undone in the opposite order in which they were performed, allowing recalculation of equivalence class sizes.) But its removal does not change the average running time, although the worst case becomes O(dn). We also force UNION to choose non-variables as equivalence class representatives whenever possible, and this modification does not affect the running time order [12].

Now consider the lattice of partitions, through which the theorem prover must search. The number of paths from bottom to a given partition may be quite large, and represents the many redundant ways in which it can be constructed. Our search procedure was written so as to traverse exactly one path to each partition, effectively converting the lattice into a search tree in which the ancestor relation entails refinement. Partitions were constructed as follows. Given n atoms in a set of formulas, we number the atoms from 1 to n. Now every block in a partition can be uniquely numbered by the lowest numbered atom it contains. Starting at bottom (the partition having all singleton blocks) we may now construct all other partitions under the following restrictions.

Whenever blocks A and B are merged to form a new (more coarse) partition:

1. B is a singleton.
2. The number of the atom in B is greater than that of the highest numbered atom in A.
3. All blocks numbered less than B will not grow at any later time.

It can easily be shown that every partition has a unique construction under these restrictions. The following algorithm summarizes the initial implemented search procedure, and is given without additional explanation for the interested reader.

```
procedure SEARCH(GROWINGBLOCK, MAXATOM)
Comment  Add to the growing block
MAXUNIFIABLE := TRUE
FOR I := MAXATOM+1 UNTIL NUMOFATOMS  DO
  IF (GROWINGBLOCK and I can be unified)
    THEN BEGIN
      MAXUNIFIABLE := FALSE
      Unify blocks GROWINGBLOCK and I
      IF SEARCH(GROWINGBLOCK,I)=success
        THEN  return (success)
        ELSE  undo the unification
      END
Comment  GROWINGBLOCK is now static
NEW := GROWINGBLOCK+1
FOR I := NEW+1 UNTIL NUMOFATOMS DO
  IF  (I is in a singleton block) THEN
    FOR J := I+1  UNTIL NUMOFATOMS  DO
      IF  (I and J can be unified)  THEN
        BEGIN
        MAXUNIFIABLE := FALSE
        Unify blocks I and J
        IF  SEARCH(I,J)=success
          THEN  return (success)
          ELSE  undo the unification
        END
IF  MAXUNIFIABLE
  N IF ( partition is contradictory)
          THEN  return (success)
          ELSE  return (failure)
  ELSE  return (failure)
end SEARCH
```

This version of the theorem prover performed well on some theorems. The following two formulas express set equivalence and the denial that set equivalence is commutative. x, s, and t are variables while a and b are constants.

```
(1) s=t<=> (ELEMENT(x,s)<=>ELEMENT(x,t))
   (2)   ¬((a=b) <=> (b=a))
```

De Champeaux [7] reported that for this example his connection graph theorem prover had generated 35 non-empty clauses when the run was abandoned. Our prover printed out the following contradictory instance of (1), (2), and a copy of (1), after running for .77 seconds (on an IBM 370-155).

```
b=a <=> (ELEMENT(x,b) <=> ELEMENT(x,a))
¬((a=b) <=> (b=a))
a=b <=> (ELEMENT(x,a) <=> ELEMENT(x,b))
```

Another good performance occurred when our prover was presented with the following unsatisfiable set of wffs:

```
(Pxyu & Pyzv & Pxvw) => Puzw
P(g(r,s), r, s)
P(1, h(1,m), m)
¬P(k(t), t, k(t))
```

The first three wffs say that we are given a set which is closed under a left associative binary operation and that there exist solutions for g and h for all equations of the form gx=y and xh=y. The last wff is the denial that there exists a right identity. In less than one second the prover discovered the unifiable contradictory partition:

```
{  {Pxyu, Pxvw, P(g(r,s),r,s)},
   {Pyzv, P(1,h(1m),m)},
   {Puzv, P(k(t),t,k(t))}  }
```

Despite these isolated encouraging results, the theorem prover became bogged down on many theorems. This was the result of at least two major deficiences. First, all maximal unifiable partitions of an atom set were being constructed, without regard to the truth-functional structure of the formulas involved; yet in general only a small fraction of these partitions were contradictory. Second, for each maximal unifiable partition constructed, the NP-complete computation of testing for boolean satisfiability had to be performed.

To reduce the number of candidate partitions, and thus alleviate both of the above problems to some extent, a heuristic was added. We demanded that every block of the partition under construction eventually contain at least two atoms of opposite "polarity" as defined in [13]. Unfortunately, performance was still unsatisfactory.

3. An almost optimal heuristic.

Consider what properties an ideal heuristic should have, in order to reduce or eliminate the two deficiencies mentioned in section 2. To eliminate the first, we would like to construct only contradictory partitions. Furthermore if that could be accomplished, then the second deficiency is also eliminated. Andrews' concept of vertical path [2], defined for formulas in negation normal form, is adapted to our prover to provide the required (completeness-preserving) heuristic.

A vertical path in a formula F, is essentially a set of literals from F that corresponds to one of the disjuncts in the disjunctive normal form of F. If F has a contradictory instance, then all vertical paths in that instance contain a pair of complementary literals.

We may construct a directed graph V[F] so that every path from an initial to terminal node in V[F] corresponds to

a vertical path in F. We call V[F] a vertical path graph (vpg), and its set of initial nodes (of indegree 0) and terminal nodes (of outdegree 0) are denoted I[F] and T[F] respectively. Node N(A) in V[F] corresponds to an occurrence of atom A in F. N(A) is positive (negative) if the corresponding occurrence of A is unnegated (negated) in the negation (or disjunctive, conjunctive) normal form of F [2], and may be written as +N(A) (-N(A)). We define V[F] inductively below. Note that in this definition a vpg is regarded as a set of nodes and ordered pairs of nodes. A is an atom, and B and C are formulas.

```
V[A] = I[A] = T[A] = {+N(A) }
V[~A] = I[~A] = T[~A] = {-N(A) }

V[B & C] = V[B]   U   V[C]   U
 {(Ni,Nj)|Ni is in T[B], Nj is in I[C]}
    I[B & C] = I[B]    T[B & C] = T[C]

V[B v C]   =   V[B]   U   V[C]
   I[B v C] = I[B] U I[C]
   T[B v C] = T[B] U T[C]
```

The remaining forms are all defined (eventually) in terms of the previous four. In these cases we give definitions for V and omit the obvious definitions of I and T.

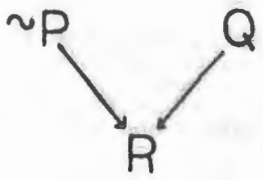
```
V[B => C]   =  V[~B v C]
V[(B <=> C)] = V((B & C) v (~B & ~C))
V[~(B & C)]  =  V(~B v ~C)
V[~(B v C)]  =  V(~B & ~C)
V[~~B]  =  V[B]
V[~(B => C)]  =  V(B & ~C)
V[~(B <=> C)] = V((~B & C) v (B & ~C))
```

As an example, consider the formula
(P => Q) & R
The reader may easily verify that

```
V[(P => Q) & R]  =
    { -N(P), +N(Q), +N(R),
      (-N(P),+N(R)), (+N(Q),+N(R)) }
I[(P => Q) & R]  =  [-N(P), +N(Q)]
T[(P => Q) & R]  =  {+N(R) }
```

We typically draw the vpg indicating node -N(P) as simply ~P. Note that the rules for the <=> connective may cause the vpg to have more than one node corresponding to a particular dag node. This does not proliferate atoms in our formulas. The several appropriate vpg nodes merely all point to the same dag node.

At the heart of the improved theorem prover is a recursive search



vpg for (P => Q) & R

procedure which conducts the search by climbing directly through the vpg. It is a backtracking procedure which seeks to force each path to contain a pair of nodes representing complementary literals (or to 'fix' each path, using Andrew's terminology [2].) To determine an order in which to attempt fixing paths, we define an order < on all paths in the vpg that begin with an initial node, called initial paths. Let I = {I1, I2, ---, Ir} be the set of initial nodes of a vpg. We assume some linear order for nodes in I so that I1<I2<,---,<Ir. We ambiguously use < for the ordering of nodes in I and the usual ordering of integers, as well as path ordering. We also assume the successors of a vpg node are ordered.

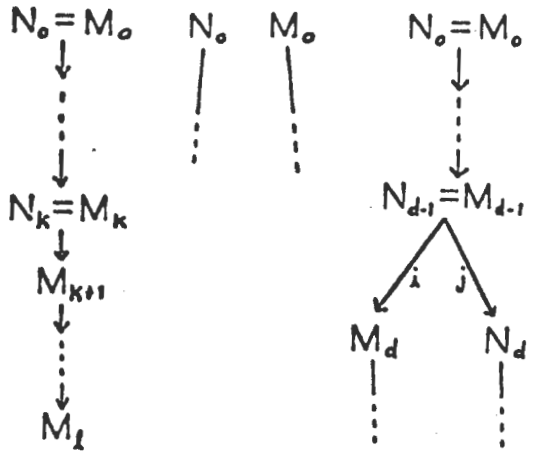
Let p1 = (n0, n1, ---, nk) and
p2 = (m0, m1, ---, m1) be initial paths in the vpg.
Let d be the least q such that nq~=mq.
We have 3 cases to consider:

Case 1: There is no such d.
Then one path is an initial segment of the other.
If k<l, then p1<p2.

Case 2: d=0
Then n0 and m0 are different nodes in I.
If n0<m0, then p1<p2.

Case 3: d>0
Then nd and md are the i-th and j-th successors of n(d-1) and m(d-1) respectively.
If i<j, then p1<p2.

Lemma 1: The relation < is a well-ordering of the initial paths of a vpg.
Proof: Transitivity is easily shown for the nine combinations of cases 1-3. It is obvious that < uniquely orders any pair of initial paths. Since we consider only finite vpgs, there are no infinite descending chains. QED

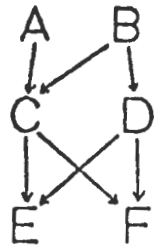


CASE 1 CASE 2 CASE 3

Example: In the vpg illustrated below, we have (BC)<(BCF), (ACF)<(BC), and (BCE)<(BCF), assuming successors are ordered left to right.

We attempt to fix paths in their defined order. However, fixing one path usually fixes many others. Consider path p=(n0,n1, ---, nk). If we fix p by unifying (the atoms represented by) nodes ni and nj where $0 \le i < j \le k$, then we have really fixed the path $(n\bar{0}, ---, nj)$ and all its extensions. We therefore would like not to consider path p until every way of fixing (n0, ---, n(k-1)) has led to failure. Furthermore we then know that the only way to fix p is by unifying nk with one of n0,n1, ---, n(k-1). So after fixing a given path p, we denote the next path to be considered by NEXTPATH(p), which is actually a defined procedure in our implementation. NEXTPATH(p) is defined as the earliest path > p which contains at least 2 nodes and is not an extension of p.

Lemma 2: Let p = (n0,n1, ---, nk) and NEXTPATH(p) = (x0,x1, ---, xr)
If p is fixed and all ways of fixing (n0,---,n(k-1)) lead to failure, then to fix NEXTPATH(p) we need only consider unifying xr with one of x0,---,x(r-1).
Proof: Let d be the earliest q such that nd~=xd.
Case 1: There is no such d.
Then since NEXTPATH(p) > p and not an extension of p, it is undefined; i.e., fixing p fixes the last path in the vpg.
Case 2: d=0
Then NEXTPATH(p) = (x0, x1).
Case 3: d>0
Then for some i, nd and xd are the i-th and (i+1)-th successors of n(d-1)=x(d-1) and d=r. So (x0,x1, ---, x(r-1)=x(d-1)) = (n0,n1, ---, n(d-1)) and all ways of fixing (n0, ---, n(k-1)) and therefore (n0, ---, n(d-1)) lead to failure. Thus fixing (x0, ---, xr) must involve xr.



NEXTPATH(BCE)=(BCF)

NEXTPATH(BC)=(BD)

NEXTPATH(BCF)=(BD)

NEXTPATH(BD)= null

Lemma 2 justifies the fact that procedure FIXPATH shown below only fixes a path p by unifying the last node in p with another node in p. If no such attempt succeeds, then FIXPATH begins work on the earliest extension of p.

```
procedure FIXPATH(PATH)
Comment  PATH is a sequence of nodes
         (n0,n1,---,nk), where k≥1
1. NEXT:=NEXTPATH(PATH)
2. IF  (PATH is already fixed)
     THEN  return (FIXPATH(NEXT))
3. FOR  i:=k-1 STEP -1 UNTIL 0 DO
     IF (unifying nk and ni fixes PATH)
       THEN BEGIN
         Unify nk and ni
         IF NEXT=() THEN return(success)
         IF FIXPATH(NEXT)=success
           THEN return (success)
           ELSE undo the unification
         END
4. IF (nk has successors)
     THEN BEGIN
       n(k+1):=first successor of nk
       APPEND(PATH, n(k+1))
       k := k+1
       Return to Step 1
       END
     ELSE  return (failure)
end FIXPATH
```

Notice that there is no guarantee that the path that FIXPATH is working on isn't already fixed. This is because fixing path p may fix many paths that are not extensions of p.

4. Current features, performance, and future work.

As it is currently implemented, the theorem prover has some advantageous

features and some weaknesses. When a path is fixed, all extensions of that path are appropriately ignored unless the path is unfixed through backtracking. The set of complementary pairs which fix a given set of paths (called a mating in [2]) could be constructed in a variety of ways. Organizing the search with path ordering prevents such redundant constructions of the same mating. Unfortunately this does not prevent non-redundant construction of different but redundant matings due to symmetries. If atoms A and B are unified, the prover may undo this unification and later unify A and B', where B' is a variant of B. The second mating cannot lead to success if the first failed. It should also be noted that different matings may induce the same partition. This is a redundancy avoided in the initial implementation.

Termination is detected whenever a path is fixed (or discovered fixed) whose extensions include the last path. This may not happen as soon as all paths are fixed, but is discovered before any crucial mating is undone.

Below is a listing of some theorems on which the program has performed fairly well. The last four theorems are not given in quantifier-free form because they double or quadruple when the quantifiers are removed. The table provides some basic performance measures. The variants column is essentially the number of copies (including the original) of the formulas required to produce a contradiction. In cases where a formula has no quantifier whose scope includes the entire formula, efficiency is improved by duplicating only outermost quantifiers rather than the entire formula [2].

```
I   (Pxyu & Pyzv) => (Pxvw <=> Puzw)
    Pg(rs),r,s  Pl,h(lm),m  ~Pk(t),t,k(t)

II  (Est <=> (Mxs <=> Mxt))
    ~(Eab <=> Eba)  (set equiv. commutes)

III  ~( ~Exists(y) All(x) Rxy <=>
          ~Exists(z) (Rxz & Rzx) )
   (Quine's version of Russell's paradox)

IV  ~( All(x) (Px <=> Exists(y) Py)  <=>
       (All(x) Px <=> Exists(y) Py) )

V   ~( Exists(x) All(y) (Px <=> Py) <=>
       (Exists(x) Px <=> All(y) Py) )

VI  ~( All(x) (Px <=> All(y) Py)  <=>
       (Exists(x) Px <=> All(y) Py) )
```

Theorem	Variants	Unif. tries	Succ. unif.	Search time (seconds)	Run time
I	1	8	4	.22	.42
II	2	27	5	.23	.36
III	2	27	9	.19	.33
IV	1	12	7	.16	.48
V	2	312	66	1.33	1.61
VI	1	11	7	.14	.48

The theorem prover could be augmented with a connection graph. We have not yet done this. The partitions of variables and terms, as well as atoms, induced by successive substitutions, are maintained as inverted trees through UNION and FIND operations. Thus unify and failure times do not increase prohibitively as the compound substitutions get larger and more complex.

The addition of the following capabilities might improve performance substantially or even dramatically. First, symmetric matings should be avoided. Each such mating doubles the search space.

A major weakness can be ascribed to the nature of backtracking search in general. Taking the wrong path near the root of the search tree may predestine failure at levels much further down. If the cause of failures could be determined, the prover might be programmed to backtrack many levels to the actual failure source. This 'multilevel backtracking' may cut out huge portions of the search tree.

Finally, no special methods for equality are currently available. An interesting problem is whether some advantage can be taken of the equivalence handling operations already present (or others like congruence closure) in order to deal with equality.

Acknowledgement

The author is very grateful to the School of Computer and Information Science and the Computer Center at Syracuse University, for providing resources which were instrumental in this research. Discussions with Bob Kowalski provided valuable insights.

This research was partially supported by The Faculty Research and Development Committee of LeMoyne College and by the National Science Foundation under Grant # MCS-8103478.

1. Andrews, P.B. Refutations by matings. IEEE Transactions on Computers 25,8 (Aug. 1976), 801-807.

2. Andrews, P.B. Theorem proving via general matings. JACM 28,2 (April 1981), 193-214.

3. Bibel, W. On matrices with connections. JACM 28,4 (Oct. 1981), 633-645.

4. Bledsoe, W.W. Splitting and reduction heuristics in automatic theorem proving. Artificial Intelligence 2 (1971), 55-77.

5. Brown, F.M. Towards the automation of set theory and its logic. Artificial Intelligence 10 (1978), 281-316.

6. Chang, C.L., and Slagle, J.R. Using rewriting rules for connection graphs to prove theorems. Artificial Intelligence 12 (Aug. 1979), 159-178.

7. de Champeaux, D. Sub-problem finder and instance checker - two cooperating processors for theorem provers. Proc. 4-th Workshop on Automated Deduction, Austin, Texas, Feb. 1979, 110-114.

8. Henschen, L.G. Theorem proving by covering expressions. JACM 26,3 (July 1979), 385-400.

9. Henschen, L.G., Lusk, E., Overbeek, R., Smith, B.T., Veroff, R., Winker, S., and Wos, L. Challange problem 1. SIGART Newsletter 72 (July 1980), 30-31.

10. Kowalski, R. A proof procedure using connection graphs. JACM 22,4 (Oct. 1975), 572-595.

11. Murray, N.V. Linear and almost-linear methods for the unification of first order expressions. Doctoral thesis, Syracuse University, July 1979.

References

12. Murray, N.V. Some observations on equivalence handling methods. IEEE Transactions on Computers C-30,5 (May 1981), 361-362.

13. Murray, N.V. Completely non-clausal theorem proving. Artificial Intelligence 18,1 (Jan. 1982), 67-85.

14. Paterson, F.S., and Wegman, M. Linear Unification. Journal of Computer Systems Science, 16 (1978), 158-167. (1960), 102-139.

15. Prawitz, D. Advances and problems in mechanical proof procedures. In Machine Intelligence 4, (Meltzer, B. and Michie, D. eds.), Edinburgh University Press, Edinburgh, 1969, 59-71.

16. Robinson, J.A. Fast unification. Presented at the Second Conference on Automated Deduction, Oberwolfach, West Germany, Jan. 1976.

17. Sickel, S. Clause interconnectivity graphs in theorem proving. IEEE Transactions on Computers, C-25 (1976), 823-834.

18. Tarjan, R.E. Efficiency of a good but not linear set union algorithm. JACM 22,2 (1975), 215-225.

19. Wilkins, D. A non-clausal theorem proving system. AISB Summer Conference Proceedings, 1974.

A LEARNING AUTOMATON THAT INFERS STRUCTURE FROM BEHAVIOR *

DIONYSIOS KOUNTANIS

WESTERN MICHIGAN UNIVERSITY

ABSTRACT

A learning system that infers the structure of a model from a finite sample of the model's behavior is described. Necessary and sufficient conditions on the sample behavior for the learning system to converge to an exact copy (up to isomorphism) of the goal structure are established. The learning system is represented as a state automaton. The automaton is a homomorphic pre-image of Menzel's learning model, which formally describes other well known theories of learning. A hierarchical decomposition of the learning automaton is derived from a hierarchical decomposition of the data that the learning automaton processes.

Key Words: Learning Automaton, Behavior, Model Structure, Homomorphic Decomposition, Convergence.

* This work was partially supported by a Fellowship Grant from the Faculty Research Fund, Western Michigan University

1. INTRODUCTION

One of the most important information processing problems is the problem of trying to choose a model to explain a collection of sample data. Amarel [1] calls this type of problem the "formation" type.

Our work here is related to this general class of problems. We are specifically concerned with the design of a learning system that infers a finite state transducer from a finite sample of input-output sequences (behavior).

Moore in his fundamental paper [9] is concerned with the question of what kind of conclusions about the structure of a finite state acceptor we can draw from external experiments. We have decomposed Moore's Experimenter into two parts, the Learning Strategy part and the Input Generation Strategy part. Suppes [11, 12] has used finite state automata as behavioral models. There are numerous researchers who have considered similar problems, such as Gold [4], Horning [5], Feldman [3] and Crespi-Reghizzi [2].

Menzel [7,8] on the other hand, has introduced state machines as models of learning. Menzel's work is systematic, rigorous and he has shown how his learning theory can describe other well known theories of learning (learning through trial and error, or through conditional reflexes, or through classification etc.)

Although much research has been done using stochastic automata as models of learning (for example Thathachar and Ramakrishnan [13]), non-probabilistic automata have hardly been used as models of learning. Scandura [10] states that the modern information processing psychology has a theoretical base which is inherently deterministic. Digital computers are also inherently deterministic devices. Non-probabilistic automata theory is a well investigated area and important tools have been developed which can be used for analysis of non-probabilistic models. The above reasons lead us to choose a non-probabilistic model for our learning system.

2. LEARNING MODEL

The structure of the learning environment we have considered is illustrated on Figure 1.

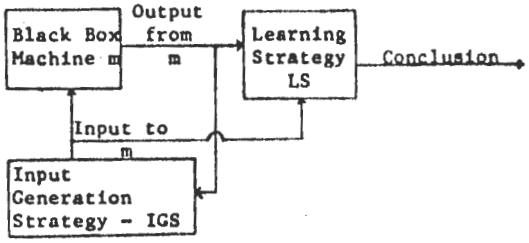


Figure 1: Learning Environment

The Black Box contains a state machine from a class m(p, q, r) of machines. The class m(p, q, r) of machines has the following properties, Kountanis [6]:

1. p is the cardinality of the set of input symbols.

2. q is the cardinality of the set of output symbols.
3. r is an upper bound on the number of states.
4. Every machine in m(p, q, r) is deterministic, completely specified and connected.

The Input Generation Strategy supplies the machine m in the Black Box with input symbols determined by the observed behavior of m.

The purpose of the Learning Strategy is to infer the machine m based on the observed input-output sequences produced by m. The LS employs a pruning technique for the inference of m.

The LS is defined as a quadruple LS=(M,J, F,m_0), where M is the set of machines that includes all machines in the class m(p,q,r) as well as all the connected submachines of every machine in m(p,q,r). M can be considered as the searching space for the LS. J=(I x O) is the set of inputs to LS, where I,O are the input, output sets over which the machines in m(p,q,r,) are defined.

F: $(M \times S) \times J \rightarrow 2^{M \times S}$ is a partial relation which describes the pruning strategy employed by the LS. S denotes the set of states of every machine in M. Intuitively F operates as follows: Assume that the LS receives an (input, output) pair from the machine m in the Black Box. Then F will generate a set of (machine, state) pairs from each machine m($m \in M$) that the LS has in its memory.
$m_0 \in M$ is the null machine, the initial machine the LS has in its memory.

The following algorithm describes the way F operates: Assume that the LS receives an (i,o) pair from the machine in the Black Box and that (m, s) is a (machine, state) pair in LS's memory.

Step 1: Does the state diagram of m have an i-transition from state s to a state s' of m?
Yes, go to step 2.
No, go to step 3.

Step 2: Is the output on the i-transition the same with o in the received (i, o) pair?
Yes, keep the machine (m, s') in memory.
No, eliminate (m, s) from the memory of the LS because the deterministic constraint on the machines has been violated.

Step 3: Does m have fewer than r states?
Yes, add a new state in m and go to step 4.
No, go to step 4.

Step 4: Generate all possible machines from m by adding a new transition from the state s to every state of m. The (i, o) pair received as input should be the label on the new transitions. Keep these machines in memory. The number of generated machines is equal to the number of m's states.

The LS applies the above algorithm to every machine in memory. The process is repeated until LS has only a single machine in memory which will be isomorphic to the machine in the Black Box. Equivalence among machines in memory is determined during the process and only one of them is kept in the memory.

Figure 3 illustrates an example of a LS. The machine m in the Black Box is from the class m(2,2,2) defined over I={a, b} and O={o, 1}. The machine m is illustrated below: We assume that the LS receives the input-output sequence σ=(a,o) (b,o) (a,1) (a,o) (a,o) from the Black Box.

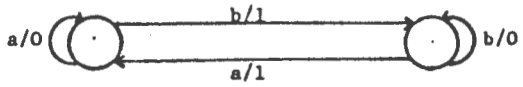


Figure 2: Machine m in the Black Box.

We have adopted the following notation:
(x,y) for the (input, output) pairs received from the Black Box.
x/y for the input/output labels on the transitions of the machines the LS conjectures.

The "heavy" lines in the illustration denote the control of the Learning Strategy.

The "light" lines denote the transitions of the machines generated by the Learning Strategy.

Note that the control lines identify the machine in the Black Box.

3. CONVERGENCE THEOREM

In this section we prove that the Learning Strategy converges to the machine in the Black Box if and only if the received sequence of input-output pairs meets certain conditions.

Let F: $(M \times S) \times J \rightarrow 2^{M \times S}$ is the partial relation defined before. We define $\overline{F}$ as the extension of F to the domain $(M \times S) \times J^*$ such that

$\overline{F}((m,s), \lambda) = \{(m,s)\}$
and
$\overline{F}((m,s), (i,o)\sigma) = \bigcup_{(m',s') \in F((m,s),(i,o))} \overline{F}((m',s'),\sigma)$

where $\sigma \in J^*$, $(i,o) \in J$, and λ is the null sequence.

We also define F' as the extension of $\overline{F}$ to the domain $2^{M \times S} \times J^*$ such that $F'(\{(m_1,s_1), \ldots, (m_k,s_k)\},\sigma) = \bigcup_{i=1}^{k} \overline{F}((m_i,s_i)\sigma)$ where $\sigma \in J^*$.

We say that the LS converges to a machine $m \in m(p,q,r)$ with respect to a sequence $\sigma \in J^*$ if and only if $F'(\{(m_o, s_o)\},\sigma) = \{(m,s)\}$, where m_o is the null machine, s_o the only state of m_o, and s a state of m.

We can easily now prove the following

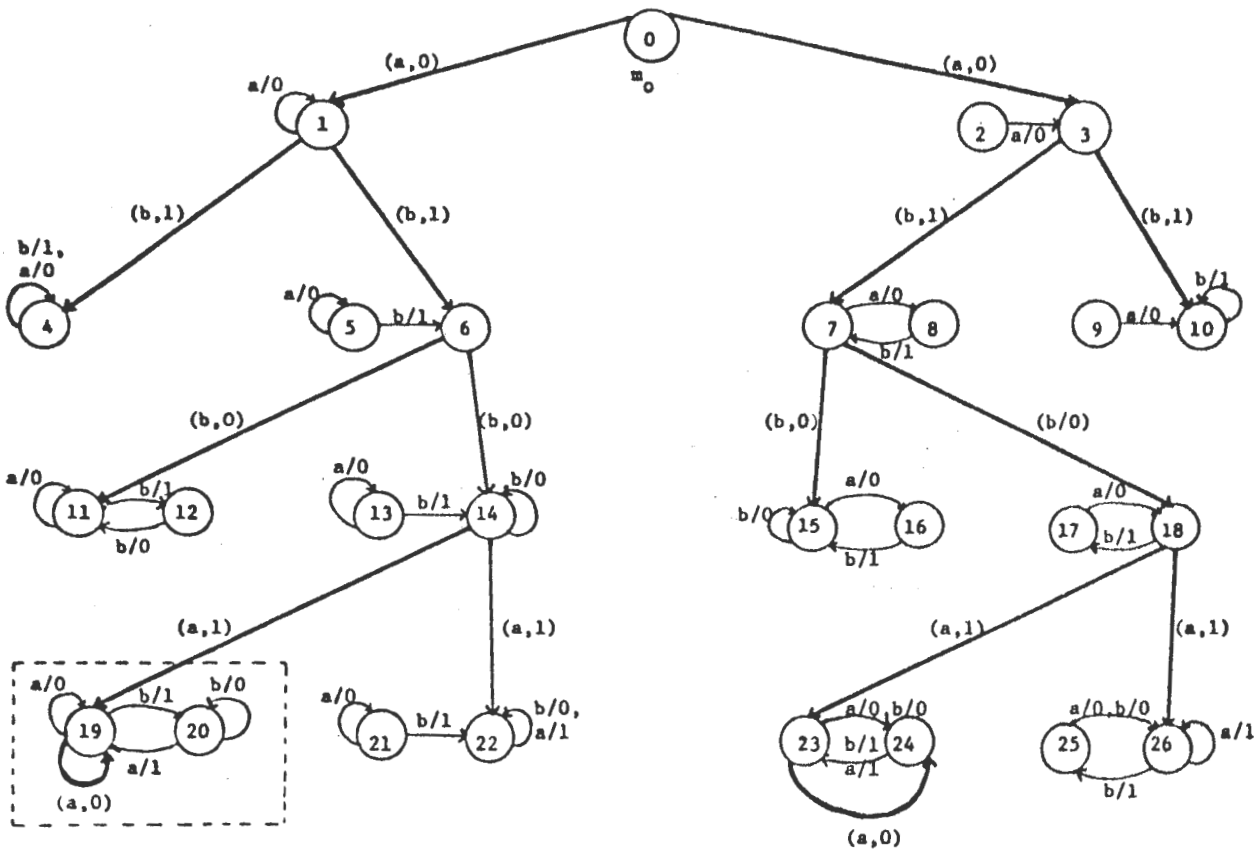


Figure 3: An Example of a LS.

convergence theorem:

Let $\sigma \in J^*$ be a sequence of input-output pairs generated by the machine m in the Black Box. Then, LS converges to m if and only if σ meets the following conditions:

1. $\sigma = \sigma_h \sigma_t$,
2. σ_h covers all the transitions of the goal machine m,
3. σ_t distinguishes (m, σ_h) from every pair $(m', s_h') \in MS_h$ where $s_h = \bar{f}_m(s_0, \sigma_h)$ with $\bar{f}_m$ being the extension of the next state function f_m of m, s_0 is the initial state of m, and MS_h is the set $MS_h = \{(n, s_h) \mid n \in m(p,q,r)$ and $s_h = \bar{f}_n(s_0, \sigma_h)$ with f_n being the next state function of n$\}$.

4. HOMOMORPHIC IMAGE OF THE LEARNING STRATEGY

We have defined the Learning Strategy as the automaton $LS = (M, J, F, m_o)$. We will now derive a homomorphism that maps LS to Menzel's Learning System.

Wolfram Menzel [8] defines his learning system in terms of functions and sets as follows:

A learning system over two finite sets I, O is a function λ such that λ: $(I \times O)^* \longrightarrow 2^{(I \times O)}$ and for each sequence $\sigma \in (I \times O)^*$, if σ is λ-admissible then $\Pi_1(\lambda(\sigma)) = I$ where $\sigma = (i_1, o_1) \ldots (i_n, o_n)$ is λ-admissible if and only if $(i_r, o_r) \in \lambda((i_1, o_1) \ldots (i_{r-1}, o_{r-1}))$ for each $r \leq$ (length of σ) and Π is the first projection function i.e. Π_1 maps a sequence σ to the set of the i-parts of σ's (i,o) pairs.

We will now define a homomorphism h that maps LS to Menzel's learning system.

Let S_m denote the set of states of a

machine m∈M. Let $I_m \subseteq (I \times O)$ denote the set of input-output pairs that appear as labels on the transitions of the state diagram of m. Then we define the following equivalence relation E among the states of machines in M:

$s_i E s_j$ if and only if $s_i \in S_m$ and $s_j \in S_n$ such that $J_m = J_n$, that is, two states are E-related if and only if they are states of machines m, n∈M which have identical sets of input-output labels on their transitions. The relation E is an equivalence relation. Thus, E defines a partition on the set S(S contains all states of every m∈M).

The blocks of the partition defined by E are $B_E(s) = \{t \mid sEt\}$. Therefore, there exists a function h: $S \rightarrow S/E$ from the set S onto the quotient set S/E whose elements are the equivalent classes defined by E. Obviously h is a homomorphism because the partition on S defined by E is a closed partition (has the substitution property).

We can now define the h-image machine M_h of the LS automaton:

$M_h = (\{B_E(s)\}, J, \delta h, Fh, B_E(s_0))$, where

δ_h: $\{B_E(s)\} \times J \rightarrow \{B_E(s)\}$ is defined as follows:

$\delta_h(B_E(s), (i,o)) = B'_E(s)$ if and only if for all $s \in B_E(s)$ we have that

$\delta(B_E(s), (i,o)) \subseteq B'_E(s)$.

F_h is the set of final states of M_h and contains every machine in m(p,q,r).

The machine M_h has the following properties:

M_h has one initial state, is completely specified over I, is connected, is completely reduced and deterministic over J=IxO.

Menzel has proven that his learning system can be represented as an automaton that has exactly the above properties.

Therefore, Menzel's learning system is a homomorphic image of our LS.

The above defined homomorphism also defines a hierarchical decomposition of the automaton LS. This decomposition has Menzel's learning system as its first component.

5. APPLICATIONS

There are seemingly endless applications using the hierarchy. The states in the finite state machine can be thought of as states of any system and the arcs from one state to another can be thought of as transitions from one state in the system to another state in the system.

For example, a medical diagnostic/treatment hierarchy can be created with the states representing certain physical condition states of a patient and the arcs representing the transition of the patients physical condition from one state to another when a certain treatment is applied. Since most medical decisions are hierarchical by nature, hierarchical data structures and processes are naturally appropriate.

Another example would be an economic system. Here, the states represent certain economic conditions (i.e. an economic state) and the arcs represent transitions from one economic condition to another when some intervention is applied. Again economic decisions are hierarchical by nature and thus hierarchical data structures and processes are appropriate.

In general for any system where hierarchical decision making occurs naturally the hierarchy is appropriate.

REFERENCES

[1] Amarel, S.: "Representations and Modelling in Problems of Program Formation," Machine Intelligence VI, (Michie and Meltzer eds.) Edinburgh University Press, 1971.

[2] Crespi-Reghizzi, S.: "An Effective Model for Grammar Inference", IFIP Congress, Yugoslavia, 1971.

[3] Feldman, J. A.: "First Thoughts on Grammatical Inference", Artificial Intelligence Memo No. 55, Computer Science Department, Stanford University, August 1967.

[4] Gold, M.: "Current Approaches to the Inference of Phrase Structure Grammars", Phrase Structure Miniconference, Montreal, Canada, 1973.

[5] Horning, J. J.: "A Study of Grammatical Inference," Technical Report No. CS 139, Computer Science Department, Stanford University, August 1969.

[6] Kountanis, D.: "Automata as Models of Learning," Computer Science Technical Report, TR-1-80, Western Michigan University, Kalamazoo, MI, 1980.

[7] Menzel, W.: "An Extension of the Theory of Learning Systems," Acta Informatica 2, Springer-Verlag, Berlin, 1974, pp. 357-381.

[8] Menzel, W.: Theorie der Lernsysteme, Springer-Verlag, Berlin, 1972.

[9] Moore, E. F.: "Gedanken-Experiments on Sequential Machines," Automata Studies, (C. E. Shannon and J. McCarthy eds.), Princeton University Press,

Princeton,N.J., pp. 129-153, 1956.

[10] Scandura, J. M.: Structural Learning I. Theory and Research, Gordon and Breach, London, 1973.

[11] Suppes, P.: "Stimulus-Response Theory of Finite Automata," Journal of Mathematical Psychology, 1969, 6, 327-355.

[12] Suppes, P.: "Stimulus-Response Theory of Automata and TOTE Hierarchies," Psychological Review, 1969, Vol. 76, No. 5, 511-514.

[13] Thathachar, M. and Ramakrishman, K.: "A Hierarchical System of Learning Automata," IEEE Transactions on Systems, Man and Cybernetics, Vol. SMC-11, No. 3, March 1981.

Data-Driven and Expectation-Driven Discovery of Empirical Laws[1]

Pat Langley
Gary Bradshaw
Herbert A. Simon
Department of Psychology
Carnegie-Mellon University
Pittsburgh, Pennsylvania 15213 USA

ABSTRACT

BACON.5 is a program that discovers empirical laws for summarizing data. The system incorporates four data-driven heuristics for relating numeric terms, recursing to higher levels of description, postulating intrinsic properties such as mass and specific heat, and finding common divisors. BACON.5 also includes expectation-driven strategies for directing search based on discoveries that the program has already made. These include heuristics for expecting similar forms of laws, reducing the amount of data that must be gathered, and taking advantage of the symmetrical form of some laws. BACON is a general discovery system that has rediscovered a number of laws from the history of physics and chemistry.

Keywords: scientific discovery, physical laws, data-driven heuristics, expectation-driven heuristics

1. Introduction

Scientific discovery is a complex, ill-defined activity, and one of the most profitable ways to study such phenomena is to construct intelligent programs that model them. In this paper we describe BACON.5, a program that discovers empirical laws for summarizing data. The core of this system is a set of general, data-driven heuristics for detecting numerical relations and proposing new terms to express these relations. However, the program also incorporates expectation-driven rules that let it take advantage of its earlier discoveries. Before moving on to describe the system in detail, we should first review some of the earlier Artificial Intelligence research on discovery, and outline the scope and limitations of the current project.

One of the earliest attempts to model scientific discovery was the simulation work of Gerwin [1]. Gerwin was interested in how humans could infer numerical laws or functions, given knowledge of specific data points. Of course, such ***descriptive discovery*** is only one part of the total scientific process. In order to understand this process, he gave subjects several sets of data and asked them to find the relationships which best summarized each data set. Using the verbal protocols collected from this task, Gerwin built a working simulation of the subjects' behaviors. The model first attempted to identify a general pattern in the data, such as a periodic trend with increasing amplitudes, or a monotonic decreasing trend. A class of functions was stored with each pattern the program could recognize; once a class was hypothesized, the system attempted to determine the specific function responsible for the data. If unexplained variance remained, the program treated the differences between the observed and predicted values as a new set of data. This procedure was used to elaborate the hypothesis until no pattern could be found in the residual data. The program also had the ability to backtrack if the latest addition to the rule failed to improve predictions. One limitation of Gerwin's simulation was that the program incorporated specific knowledge about the shapes of functions within a specified range. Therefore, these functions could not have variable parameters associated with them. Even though Gerwin's model could only solve a very restricted range of problems, it was an important step in understanding the discovery process.

Another early discovery system was DENDRAL [2], a program that identified organic molecules from mass spectrograms and nuclear magnetic resonances. The system identified chemical structures in three main stages – planning, generating plausible structures, and testing those structures. The first stage used patterns in the data to infer that certain familiar molecules were present. Considering these molecules as units drastically reduced the number of structures produced during the generation stage. This second phase used knowledge of valences, chemical stability, and user-specified constraints to generate all plausible chemical structures. In the final testing stage, the system predicted mass spectrograms for each of these structures, which were then ranked according to their agreement with the data. DENDRAL relied on considerable domain-specific knowledge, which was laboriously acquired through interaction with human experts in organic chemistry.

In order to reduce their dependence on human experts, the same researchers designed META-DENDRAL [3], a system that acquired knowledge of mass spectroscopy which could then be used by the DENDRAL program. META-DENDRAL was provided with known organic compounds and their associated mass spectrograms, from which it formulated rules to explain these data. Two types of events were used to explain spectrograms – ***cleavages*** in the bonds of a molecule and ***migrations*** of atoms from one site to another. Although plausible actions were determined using domain-specific chemical knowledge, the conditions on rules were found through a much more general technique [4]. META-DENDRAL has successfully discovered new rules of mass spectroscopy for three related families of organic molecules.

Lenat [5] has described AM, a system that has rediscovered important concepts from number theory. The program began with some 100 basic concepts such as ***sets***, ***lists***, ***equality***, and ***operations***, along with some 250 heuristics to direct the discovery process. These heuristics were responsible for filling the facets of concepts, suggesting new tasks, and creating new concepts based on existing ones. New tasks were ordered according to their ***interestingness***, with tasks proposed by a number of different heuristics tending to be more intersting than those proposed by a single rule. Using this measure to direct its

[1]This work was supported in part by ONR Contract Number N00014-82-0168, and in part by NIMH Grant MH-07722.

search through the space of mathematical concepts, AM defined concepts for the integers, multiplication, divisors-of, prime numbers, and the unique factorization theorem. Like META-DENDRAL, Lenat's system incorporated some very general strategies, as well as some domain-specific knowledge about the field of mathematics.

In our work on BACON, we have attempted to develop a general purpose descriptive discovery system. Rather than relying on domain-dependent heuristics, as many of the earlier discovery systems have done, BACON incorporates weak yet general heuristics that can be applied to many different domains. The current version addresses only the descriptive component of scientific discovery. It does not attempt to construct explanations of phenomena, such as the atomic theory or the kinetic theory of gasses, but we will have more to say on this in a later section. Neither is the system meant to replicate the historical details of various scientific discoveries, though of course we find those details interesting. Instead, it is intended as a model of how discoveries *might* occur in these domains.

Descriptive discovery may take either of two basic forms: one may start from the data and use very general strategies to uncover regularities in those data; or one may bring certain expectations to the task and examine the data to see if they match those expectations. Earlier versions of BACON [6, 7, 8] relied entirely on *data-driven* discovery methods. The current version takes advantage of these heuristics, but also incorporates a number of *expectation-driven* discovery techniques. The latter take advantage of discoveries that have already been made to direct and simplify the search process in new situations. We have chosen to organize the paper around the system's discovery methods. Since the expectation-driven heuristics work with the results of the data-driven approaches, we will begin by focusing on the data-driven components and then move on to their expectation-driven counterparts. Both types of heuristics are implemented as condition-action rules in Forgy's [9] OPS4 production system formalism.

2. Discovering Numeric Relations

BACON.5's most basic heuristic attempts to discover polynomial relations between two variables that take on numeric values. This rule computes the successive derivatives of one term with respect to the other, until it arrives at a set of constant values. The level of the constant derivative tells BACON the highest power necessary in the polynomial it seeks, while the constant determines the coefficient of this term. As in Gerwin's system, this component is subtracted out, and the technique is repeated on residual values. This process continues until all of the variance has been accounted for, and the program has determined the complete functional relation between the two variables.

X	Y	Y'	Y''
1	6		
		14	
3	34		3
		29	
6	121		3
		50	
10	321		3
		77	
15	706		

Table 1. Determining the coefficient of a quadratic term.

As an example, let us consider BACON.5's use of this heuristic to discover the law $y = 3x^2 + 2x + 1$. The program begins by examining values of the dependent term y for different values of the independent term x, as shown in Table 1. Since y is not constant, the system computes the values of y', the first derivative with respect to x. In the table, the first value of y' is $(34 - 6)/(3 - 1) = 14$, while the second value is $(121 - 34)/(6 - 3) = 29$. Since these values are not constant either, BACON examines the second derivative y'', basing its computation on the values of y' and x. Thus, the first value of y'' is $(29 - 14)/(6 - 1) = 3$, while the second is $(50 - 29)/(10 - 3) = 3$. In this case, the program finds the constant value it seeks; this tells BACON that an x^2 term is present in the final law, and that its coefficient is 3.

However, more remains to be done before the discovery is complete. After subtracting out the $3x^2$ term, BACON attempts to relate the values of $y - 3x^2$ to the independent term x, as shown in Table 2. This time the first derivative is the constant 2, implying that an x term with a coefficient of 2 is also present in the final law. Subtracting this new component out as well, the constant value 1 immediately results, as we see in Table 3. BACON.5 includes this value as the final term in the law it has discovered, $y = 3x^2 + 2x + 1$, which completely summarizes the original set of observed data.

X	$Y - 3X^2$	$(Y - 3X^2)'$
1	3	
		2
3	7	
		2
6	13	
		2
10	21	
		2
15	31	

Table 2. Determining the coefficient of a linear term.

This method lets BACON.5 discover any of a large class of functions that can be expressed as polynomials with integer powers and real coefficients. In cases where no polynomial can be found, the system considers various powers of the dependent term, so that an even larger set of relations can be discovered. Thus, BACON can uncover relations such as $y^2 = 6.71x^3 + 4.23x$ and $y^{-1} = 3.5x^2$. The system entertains only one hypothesis at a time, and since simpler relations are considered before more complex ones, they are preferred if they are found to hold.

X	$Y - 3X^2 - 2X$
1	1
3	1
6	1
10	1
15	1

Table 3. Determining the constant term in an equation.

3. Levels of Description

By itself, the above differencing heuristic can discover numeric relations between *two* variables, but more complex relations lie beyond its scope. In order to find laws relating many terms, BACON.5 invokes a second data-driven heuristic that lets it summarize regularities at different *levels of description*. Upon discovering a law at one level, this method stores the coefficients from that law at the next higher level. Once enough of these higher level values have been gathered, BACON attempts to relate them to the independent term that was varied in each of the experiments. The system employs the same differencing technique to find the second level law as it did at lower levels. After a law at the second level has been found, the program recurses to still higher levels, until all of the data have been summarized.

BACON.5's discovery of the ideal gas law provides a useful example of this strategy. This law may be stated as $PV = 8.32N(T - 273)$, where P is the pressure on a quantity of gas, the dependent term V is the volume of the gas, T is the temperature of the gas in degrees Celsius, and N is the quantity of gas in moles. In uncovering this law, BACON first finds the relation $V^{-1} = aP$, where a is a parameter that varies with different values of T and N. Upon comparing the values of a and T, the system finds the law $a^{-1} = bT + c$, where b and c represent second level parameters that potentially vary with N. Finally, the program finds that $b = dN$, and that $c = eN$. Substituting these relations into the first law, we arrive at the equation $V^{-1} = P(dNT + eN)^{-1}$. BACON.5 calculates the value of d to be 8.32, and e to be -2271.36. When the factor 8.32 is divided out, e becomes -273, or the absolute zero point expressed in the Celsius scale. Thus, the equation is equivalent to the standard form of the ideal gas law. Table 4 summarizes the steps taken in this discovery, comparing BACON's version of the law with the standard version, and showing the independent terms held constant at each level of description.

BACON'S VERSION	STANDARD VERSION	CONSTANT TERMS
$1/V = aP$	$PV = K$	T,N
$1/V = P/(bT+c)$	$PV = K(T-273)$	N
$1/V = P/(dNT + eN)$	$PV = 8.32N(T-273)$	

Table 4. Summary of Ideal Gas Law Discovery.

Taken together, the heuristics for relating numeric terms and recursing to higher levels give BACON.5 considerable power. Using these two strategies, the system has successfully rediscovered versions of Coulomb's law of electrical attraction, Kepler's third law of planetary motion, and Ohm's law for electrical circuits. Table 5 presents the forms of these laws, along with that for the ideal gas law. Variables are shown in upper case, while coefficients are given in lower case. Superficially, the equations in the table have quite different forms, yet all can be expressed as combinations of the polynomial relations for which BACON searches.

Ideal gas law	$PV = rNT$
Coulomb's law	$F = aQ_1Q_2/D^2$
Kepler's third law	$D^3/P^2 = k$
Ohm's law	$v = rI + IL$

Table 5. Numeric laws discovered by BACON.5.

4. Postulating Intrinsic Properties

The heuristics we have discussed so far are fine for relating numeric terms, but they are of little use when an independent term takes on *nominal* or *symbolic* values. In such cases, BACON.5 draws on a third data-driven heuristic that postulates *intrinsic properties*. This rule associates the values of the numeric dependent term with the nominal independent values, and retrieves them in later situations. In this context, BACON moves beyond the relatively simple process of curve fitting, and takes on some features of explanatory discovery.

For example, consider a version of Ohm's experiment in which the batteries and wires take on nominal values, so that one can distinguish between them but measure none of their characteristics. Ohm's law may be stated as $I = V/R$, where I is the current flowing through a circuit, V is the voltage associated with a wire, and R is the resistance of the wire. (We assume here that the internal resistance is negligible.) Table 6 presents data that might be gathered in an experiment with three batteries (A, B, and C) and three wires (X, Y, and Z). The values of the current were calculated on the assumption that $V_A = 4.613$, $V_B = 5.279$, $V_C = 7.382$, $R_X = 1.327$, $R_Y = 0.946$, and $V_Z = 1.508$.

BATTERY	WIRE	CURRENT	CONDUCTANCE	SLOPE
A	X	3.4763	3.4763	1.0
A	Y	4.8763	4.8763	1.0
A	Z	3.0590	3.0590	1.0
B	X	3.9781	3.4763	1.1444
B	Y	5.5803	4.8763	1.1444
B	Z	3.5007	3.0590	1.1444
C	X	5.5629	3.4763	1.6003
C	Y	7.8034	4.8763	1.6003
C	Z	4.8952	3.0590	1.6003

Table 6. Postulating the property of conductance.

Focusing on the first three rows of this table, BACON.5 finds that with the battery set to A and varying the wire, the current of the circuit varies as well. Since it cannot apply its numeric heuristic in this situation, the program proposes *conductance* as an intrinsic property of the wire, and bases the values of this new term on those of the current. Having done this, BACON can apply its differencing heuristic, and finds a linear relation between the current and the new property, with a slope of one. Of course, this is hardly surprising, since the conductance was defined so that this relation would hold.

However, upon varying the values of the battery, BACON retrieves the same values of the conductance in the new situations, as shown in the fourth through ninth rows. When these are compared to the currents, the system discovers other linear relations with different slopes. After recursing to a higher level of description, BACON uses these new parameters to postulate an intrinsic property associated with the battery, which we would call the *voltage*. The retrieval technique is actually stated as a separate heuristic, and shows more similarity to the expectation-driven heuristics we shall discuss later than to the data-driven ones. We have mentioned it here because the data-driven process of postulating an intrinsic property has little purpose without the ability to retrieve the associated values at later times.

Unfortunately, the discovery of intrinsic properties is more complex than we have made it appear. Some properties exist which are associated not with one, but with many, nominal terms. An obvious example is the coefficient of friction, which is a function of *pairs* of surfaces. To avoid difficulties in such cases, BACON.5 takes a conservative path by comparing different sets of intrinsic values. If a linear relation is found, the system generalizes and retrieves values as in the Ohm's law example. However, if no relation is found, it retains the additional conditions. Table 7 lists some of the laws rediscovered by BACON.5 that incorporate intrinsic properties. These include a version of Archimedes' law of displacement, in which the system computes the volumes of irregular solids as well as their density, and Proust's law of definite proportions, in which a constant weight ratio is associated with an element-compound pair.

Ohm's law	$v = ri$
Archimedes' law of displacement	$d = W/v$
The law of definite proportions	$k = W_e/W_c$

Table 7. Laws discovered with intrinsic properties.

5. Finding Common Divisors

The history of chemistry from 1800 to 1860 provides some additional examples of the discovery of intrinsic properties, with an interesting complication. In 1808, John Dalton set forth the *law of simple proportions*, which stated that when two elements could combine to form different compounds, the weights contributed by one element for a constant weight of the other always occurred in *small integer proportions* to each other. In 1809, Joseph Gay-Lussac found evidence for his *law of combining volumes*, which stated that a similar relation held for the relative volumes contributed by gaseous elements in chemical reactions. Again, in 1815, William Prout noted that the atomic weights of the known elements were all very nearly divisible by the weight of hydrogen. And finally, in 1860, Stanislao Cannizzaro pointed out that when a given element took part in different reactions, the ratios of the element's weight and the volume of the resulting compound always occurred in small integer proportions.

ELEMENT	COMPOUND	W_e/v_c	INTEGER	DIVISOR
HYDROGEN	WATER	0.0892	2.0	0.0446
HYDROGEN	AMMONIA	0.1338	3.0	0.0446
HYDROGEN	ETHYLENE	0.0892	2.0	0.0446
OXYGEN	N_2O	0.715	1.0	0.715
OXYGEN	SO_2	1.430	2.0	0.715
OXYGEN	CO_2	1.430	2.0	0.715
NITROGEN	N_2O	1.250	2.0	0.625
NITROGEN	AMMONIA	0.625	1.0	0.625
NITROGEN	NO_2	0.625	1.0	0.625

Table 8. BACON.5's rediscovery of Cannizzaro's law.

BACON.5 incorporates a fourth data-driven heuristic that enables it to discover these regularities in the chemical data. When the system is about to postulate a new intrinsic property, this rule examines the dependent values to see if they have a common divisor. If none can be found, then the process continues as described in the last section. However, if the numbers can be evenly divided, then the resulting integers are used as the intrinsic values instead of the original numbers. Also, the common divisor is associated with the terms that were held constant, instead of the *1.0* that would normally be used. This means that even in cases where BACON.5 cannot generalize and so retrieve a set of intrinsic values in a new situation, the common divisors let the system break out of the tautological circle and make further interesting discoveries.

Table 8 summarizes BACON.5's reformulation of Cannizzaro's discovery. The system is given control over two independent nominal terms - one of the elements entering into a reaction, and the resulting compound. The dependent variable is w_e/v_c, or the weight of the element used in the reaction, divided by the volume of the compound that results. For the element *hydrogen*, different compounds lead to different values of w_e/v_c, so the system postulates an intrinsic property. However, the dependent values are all divisible by *0.0446*, so the integers *2*, *3*, and *2* are used as the intrinsic values instead of the originals. This process is repeated with the elements *oxygen* and *nitrogen*, but in these cases the divisors *0.715* and *0.625* are found instead. The integers in the table correspond to the coefficients on the given elements in the balanced equations for each reaction, while the divisors correspond to the relative atomic weights of the elements. When these divisors are carried along to the next level of description, BACON.5 also notes that they can all be divided by the value associated with *hydrogen*; this statement is a variant on Prout's hypothesis. By searching for common divisors, BACON has replicated some of the major empirical discoveries of nineteenth century chemistry.

6. Expecting Similar Relations

We have now completed our survey of BACON.5's data-driven heuristics. The remainder of the system's strategies draw upon information gathered in this bottom-up manner to reduce search at later stages. Thus, when we speak of expectation-driven heuristics, we do not mean to imply that BACON starts with knowledge of a particular domain. Rather, we mean that the program is capable of taking advantage of discoveries it has made at early stages to simplify this process at later points.

The simplest of these heuristics proposes that if BACON.5 has found a law in one context (i.e., when certain variables are held constant), it should expect a similar *form* of law to hold in a new context (i.e., when those terms take on different values). For example, this *similar relations* heuristic could be used after the system has discovered Kepler's third law for the planets orbiting the sun, to predict an analogous law to hold for the moons of Jupiter. Specifically, if the law $D^3 = 1.0P^2$ were found in the first situation, BACON.5 expects that a law of the form $D^3 = kP^2$ would hold in the new case, though it would not yet know the value of the parameter k. Such a prediction allows BACON to replace its search through the space of possible relationships between two variables with a simple calculation designed to test the expected relationship. If this relationship holds, BACON calculates the values of the unknown parameters and moves on to further discoveries.

Previous versions of BACON always utilized the same number of observations to find relationships between variables in its experiments. However, once the system expects a particular form of a law to hold, it can determine the number of observations necessary to estimate the desired parameters. Using this *data reduction* heuristic, BACON only collects the minimum number of observations necessary to complete its description of the current law. If D were being expressed as a function of P in the above example, BACON.5 would need only

three data points to determine the value of k for the Jovian moons.[2]

Taken together, these two heuristics significantly reduce the program's search through both the space of data and the space of rules. The actual amount of savings depends on the number of superfluous data points. In order to evaluate the impact of the new heuristics, BACON was given six values of each independent variable in four separate discovery tasks. Performance of the purely data-driven system was compared to systems incorporating the expectation-driven heuristics, and is shown in Table 9. From this table, it can be seen that the similar relation heuristic only resulted in a small amount of savings. This result is somewhat misleading, because the amount of search required by the differencing technique was significantly reduced; however, the OPS4 interpreter was slowed by the inclusion of an additional condition-action rule, so the effect was masked. For more complex forms of laws, the computational savings would be greater.

	DD	DD + SR	DD + SR + DR
IDEAL GAS LAW	35	34	21
COULOMB	35	35	23
OHM	3	3	3
KEPLER	3	3	3

Table 9. Time to discover numeric laws in CPU seconds.[3]

The present system employs a few simple heuristics for dealing with noise. In executing the differencing technique, BACON.5 checks the current derivative term to see if its values are constant. All values which fall within a small interval of one another are accepted as equivalent. The program also calculates the number of outliers, or *exceptions* to the current relationship. If the number of exceptions is a small proportion of the total number of data points, BACON.5 decides the current term is constant, and updates its functional description. Although these methods allow BACON.5 to cope with modest amounts of noise, more sophisticated techniques might be required to deal with very noisy data.

One such technique might be to check the dependent term for systematic trends. The values of y in Table 1 are monotonically increasing, for example, which suggests a higher order derivative should be calculated. If no such trends were found, BACON.5 could accept the current relationship, even though the number of outliers was large. A second technique would be to allow the program to store several possible relationships between the current independent and dependent terms. Beam searching techniques could be used to limit the number of competing hypotheses BACON entertained at any given time, and the program could design critical experiments to determine the best description of the data. Finally, if the system discovered promising relationships in parts of the data, the expectation-driven heuristics discussed above could help BACON to develop a consistent interpretation of the data, even in the presence of substantial noise. Combining these techniques should allow BACON to deal with realistic amounts of noise in data in a robust manner.

[2] Of course, more would be required if significant noise were present, but the principle of reduced data would remain.

[3] DD = data-driven heuristics, DD + SR = data-driven and similar relation heuristics, DD + SR + DR = data-driven, similar relation, and data reduction heuristics.

7. Discovering Symmetrical Laws

The assumption of symmetry has been a powerful aid in the discovery of physical laws. Table 10 presents three well-known laws that exhibit symmetry. Although BACON.5 could discover these laws without any heuristics other than those we have already described, the inclusion of a new component that *postulates* symmetry significantly reduces the search required to find these laws. This new heuristic waits until all the terms associated with an object have been related, and then *assumes* that the same relation will hold for a second set of terms that are associated with an analogous object. The resulting complex terms are then combined into a symmetrical law.

Snell's law of refraction	$\sin\theta_1/n_1 = \sin\theta_2/n_2$
Conservation of momentum	$m_1(V_1 - U_1) = -m_2(V_2 - U_2)$
Black's specific heat law	$c_1M_1(T_1 - F_1) = -c_2M_2(T_2 - F_2)$

Table 10. Symmetrical laws discovered by BACON.5.

As an example, consider BACON.5's discovery of Snell's law of refraction, as summarized in Table 11. The program starts with two objects and two variables associated with each object - the *medium* through which light passes, and the *sine* of the angle the light takes. Varying $medium_2$ and holding $medium_1$ and $sine\theta_1$ constant, the system postulates an intrinsic property, n_2, whose values are associated with different media. Of course, the ratio $\sin\theta_2/n_2$ has the constant value *1.0*. At this point, BACON.5 relates the terms associated with the second object, and decides that it should examine the values of $\sin\theta_1/n_1$ and relate them to the former ratio. Upon gathering additional data, the program discovers that the two ratios are identical, or that $\sin\theta_1/n_1 = \sin\theta_2/n_2$, which is one statement of Snell's law.

$MEDIUM_1$	$\sin\theta_1$	$MEDIUM_2$	$\sin\theta_1$	N_2	$\sin\theta_2/N_2$
VACUUM	0.25	WATER	0.33	0.33	1.0
VACUUM	0.25	OIL	0.37	0.37	1.0
VACUUM	0.25	GLASS	0.42	0.42	1.0

Table 11. Discovering Snell's law of refraction.

The BACON.5 system has discovered two other symmetrical laws - conservation of momentum and Black's specific heat law - following very similar paths. Table 10 presents the full form of the laws; directly observable terms are shown in upper case, while intrinsic properties are shown in lower case. The program has also discovered two different versions of Joule's law of energy conservation, using a simple form of reasoning by analogy. This strategy states that if the same set of terms occurs in more than one experiment, one should consider combining them in the same fashion as proved useful before. For a more complete description of this heuristic and its application to Joule's law, the reader is directed to an earlier article on BACON [10].

In summary, we have seen that BACON's expectation-driven heuristics - expecting similar relations, reducing the data that is gathered, and postulating symmetrical laws - allow it to discover empirical laws with considerable reduction in search. Actual computational savings for three symmetric laws are shown in Table 12. From this table, it can be seen that, when combined, BACON.5's expectation-driven heuristics result in

major savings. Moreover, these heuristics accomplish this with little loss in generality, since relations such as symmetry can be found in a wide variety of scientific domains.

	DD	DD + SR + DR	DD + SR + DR + SY
MOMENTUM	515	212	8
SNELL	40	40	8
BLACK	8433	2200	23

Table 12. Time to discover symmetric laws in CPU seconds.[4]

8. The Importance of Structure

In the previous sections, we have described the empirical discovery system BACON.5. Given a set of numeric or nominal variables, this system employs a number of heuristics to determine the relation between those terms. Yet it is worth noting that the most interesting of BACON's heuristics address aspects of discovery that lie beyond the simple relation of variables. For example, when an intrinsic property is postulated, it is always associated with some *object* or class of objects. Similarly, the symmetry and analogy heuristics apply only in situations where the same terms are associated with different objects. (In the symmetry case, identical terms are attached to different objects *within* an experiment, while in the analogy case the identity falls *across* experiments.)

In summary, these heuristics appear to incorporate some notion of *structure* which extends beyond the simple variable-value representation used in BACON.5. Given this view, one drawback of BACON is that it represents this structure *implicitly* rather than *explicitly*. Thus, in replicating Ohm's experiment, the program is told about the battery, the length of the wire, and the current, but it does not understand that the wire must be connected to the battery to generate the current. Similarly, in the conservation of momentum experiment, BACON is given variables for the objects along with their initial and final velocities; however, it is unaware that the initial velocities are transformed into the final velocities by a collision, and that if no collision occurs, the velocities will remain unchanged. In other words, BACON.5 attempts to discover quantitative laws before it has mastered the *qualitative laws of structure* [11]. This feat can be accomplished, but only if the system is presented with a set of variables that have been carefully selected to contain those qualitative relationships.

Future versions of BACON should represent structural relations explicitly, and should attempt to discover the qualitative laws of a situation (e.g., that objects collide and change direction, or that some chemicals combine to form new chemicals) before moving on to considering quantitative laws. Such an approach would be much more consistent with historical developments in science than the current implementation. Moreover, once the system has arrived at a structural model for a situation, this model may find another use as an *explanation* for a quantitative law found in some other situation. This is an important point, since many explanatory theories - including the atomic theory, the kinetic theory of gasses, and the germ theory of disease - are primarily structural models. Thus, by exploring the role of structure in a descriptive discovery system like BACON, we may come to a fuller understanding of explanatory science as well.

For instance, consider the kinetic theory of gasses, which can be used to explain the ideal gas law. Central to the kinetic theory is the notion of colliding molecules that obey conservation of momentum. The hypothesis that a gas is composed of microscopic objects (similar to their macroscopic counterparts in the momentum experiment) provides an explanation of the macroscopic relation between temperature, volume, and pressure. We do not claim to fully understand the process by which such explanations are constructed, though some form of reasoning by analogy seems a likely candidate. In any case, the relation between qualitative laws of structure and explanation is a promising direction for future research.

It is interesting to note that one of BACON's current heuristics - searching for common divisors - could play an important role in such an explanatory discovery system. This results from the fact that the existence of a common divisor for a set of data suggests an important structural aspect of those data, namely that the objects involved in the experiment consist of quanta. Thus, one can imagine an extended version of BACON that, upon finding common divisors in chemical reactions, would invoke a prototype atomic theory to explain this fact.

Finally, we should note that an emphasis on qualitative laws of structure may provide a new approach to the dual problems of noise and irrelevant variables. Given an understanding of the structure of some situation, it may be possible to eliminate some relationships and some variables even before any quantitative data are gathered. For example, given the principle "no action at a distance" and an experimental context in which two objects never touch or even approach each other, one can immediately predict that the variables associated with these objects will be unrelated. Again, this is an area in which our ideas remain vague, but it is also an area that deserves further attention.

9. Conclusions

In this paper we have described BACON.5, an empirical discovery system that draws on data-driven heuristics for finding numeric relations between two variables, recursing to higher levels of description, postulating intrinsic properties, and finding common divisors. In addition to its data-driven techniques, BACON also incorporates expectation-driven heuristics for expecting similar relations, reducing the amount of data that must be gathered, assuming symmetrical laws, and reasoning by a simple type of analogy. These latter rules take advantage of discoveries BACON has made itself instead of drawing on knowledge about some particular domain. Thus, the program retains considerable generality, as evidenced by the broad range of laws it has been able to discover. In addition, the expectation-driven methods reduce the overall search that BACON must perform in discovering a law.

We have also seen that some of BACON's heuristics incorporate a notion of structure, but that this knowledge is represented implicitly. Future versions of the system should represent structural information explicitly, and attempt to discover qualitative laws before moving on to quantitative ones. This approach should provide new methods for handling noise and determining relevant variables, but it may do more than simply improve BACON's techniques for discovering descriptive laws. We hope that a concern with qualitative laws of structure will shed light on the process of explanatory discovery as well.

[4] DD = data-driven heuristics, DD + SR + DR = data-driven, similar relation, and data reduction heuristics, DD + SR + DR + SY = data-driven, similar relation, data reduction, and symmetry heuristics.

References

[1] Gerwin, D. G. Information processing, data inferences, and scientific generalization. *Behavioral Science*, 1974, *19*, 314-325.

[2] Feigenbaum, E.A., Buchanan, B.G., and Lederberg, J. On generality and problem solving: A case study using the DENDRAL program. *Machine Intelligence 6*. Edinburgh University Press, 1971.

[3] Buchanan, B. G., Feigenbaum, E. B., and Sridharan, N. S. Heuristic theory formation: Data interpretation and rule formation. In D. Michie (ed.), *Machine Intelligence 7*. New York: American Elsevier, 1972, 269-290.

[4] Mitchell, T. M. Version spaces: A candidate elimination approach to rule learning. *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, 1977, 305-310.

[5] Lenat, D. B. Automated theory formation in mathematics. *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, 1977, 833-842.

[6] Langley, P. Data-driven discovery of physical laws. *Cognitive Science*, 1981, *5*, 31-54.

[7] Bradshaw, G., Langley, P., and Simon, H. A. BACON.4: The discovery of intrinsic properties. *Proceedings of the Third National Conference of the Canadian Society for Computational Studies of Intelligence*, 1980, 19-25.

[8] Langley, P., Bradshaw, G., and Simon, H. A. Rediscovering chemistry with BACON.4. To appear in J. Carbonell, R. Michalski, T. Mitchell (eds.), *Machine Learning*.

[9] Forgy, C. L. The OPS4 reference manual. Technical report, Department of Computer Science, Carnegie-Mellon University, 1979.

[10] Langley, P., Bradshaw, G. L., and Simon, H. A. BACON.5: The discovery of conservation laws. *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, 1981, 121-126.

[11] Newell, A. and Simon, H. A. Computer science as empirical enquiry: Symbols and search. *Communications of the ACM*, 1975, *3*, 113-126.

NON-TEMPORAL PREDICTION - A DISTRIBUTED SYSTEM FOR CONCEPT ACQUISITION

PAUL ROBERTSON

UNIVERSITY OF TEXAS AT DALLAS

Abstract

A theory of learning based on the principle of non-temporal prediction has been developed. The significant aspects of the theory are that a data-driven part of learning is isolated and its computation demonstrated in the form of a highly distributed network of simple processors. Implementional considerations give rise to a number of architectural constraints that suggest a particular processor topology. An outline of the theory is presented.

Keywords- Models of Learning, Skill Acquisition Distributed Problem Solving, Society of Mind.

Introduction

Traditional studies of learning have concentrated on situations involving a 'teacher' (often in the form of a timely reward). Explaining the learning phenomenon has usually depended on the notion of recency, whereby a memory trace decays with time. The notion of recency brings with it some serious problems, furthermore, it is possible to explain the observed phenomenon without reference to trace decay (Robertson [13]). It is contended that although time plays an important role in many learning situations, there is another more primitive mechanism for learning that is time invariant - Non-Temporal prediction.

Knowledge is viewed as a means of predicting events in the world. Our survival is in a large part dependant on our ability to 'predict' the world. Learning is seen as a mechanism that has evolved to meet this need. Predictions about the world can be divided into time-related (temporal) predictions and predictions that involve classifying events. The latter form of prediction is time invariant, learning of this nature will be refered to as 'learning through non-temporal prediction. Two recent trends have been particularly successful within certain areas of artificial intelligence, especially that of low level vision. The first has been a concentration on what information can be extracted from a scene (usually visual), knowledge free, that is without top-down influence. Notable successes with this approach are those of Marr [1 2 3] as well as some recent work on parallel algorithms for extracting feature points in optical flows. The other important trend (not entirely disjoint from the first) has been a drift away from the Von-Neumann architecture as the default for modelling cognitive processes. By choosing an appropriate architecture, many of the traditional problems disappear, and the issues that underlie the problem computationally surface. This is particularly evident in Fahlman [4] where a highly distributed architecture is used to implement a semantic network and marker propagating scheme. A class of problems that are definable as the union and intersection of sets can be computed at high speed where an expensive search would otherwise have been needed. Having the wrong architecture can render an essentially simple task restrictively costly. Instead of devoting a lot of effort to finding ways of speeding up these algorithms, it is almost certainly better to direct the effort towards finding architectures well suited to the task.

Progress in VLSI techniques along with the emergence of highly distributed architectures, has awakened an interest in examining what can be done with certain architectures based on simple 'neuron like' processors, such as Hinton [5]. Computational intuitions for this kind of architecture are presently lacking. A few pencil and paper exercises have been attempted with some success such as Marr [6], Minsky [7 8], Abelson [9], Waltz [10]. There have also been a number of successful implementations such as some work on relaxation algorithms. In this paper we concentrate on two aspects of such architectures. First, how natural constraints can be brought to bear on the problem of choosing suitable categories without the help of a teacher (without top down control), and how this forms the basis for a low-level theory of learning. Second, an attempt to constrain the N^2 explosion of processor interconnections based on computational intuitions and physiological evidence.

Work on learning has concentrated on learning that involves a teacher to guide the learning process. Winston [16] stressed the importance of the role of the teacher. We argue in this paper that the importance of the teacher was exaggerated for the following reasons. The teacher according to Winston, is responsible for the choice of example/near miss in two respects. First that the examples form a natural sequence,

each building upon what the previous one established. Second that the differences between the example presented and the program's model should all be 'significant' differences. The sequence of exposure is clearly important, it is a consequence of the structural nature of the concept being taught. The very rigid nature of Winston's Near Miss is a consequence of the 'architecture of the program'. Near miss was being used as a form of control. Children learn many complex concepts without the aid of a teacher, such as language. The real world presents examples randomly, we are not told which are near misses and which are examples, furthermore the order is not choosen so as to form a sequence. The architecture presented in this paper explains how near miss can be inferred by exploiting natural constraints, and how examples that are not near misses (premature in the natural sequence of exposures) can be ignored. Nature must provide examples more than once in order that by selectively ignoring, the correct sequence can be extracted.

Teacher viewed as a resource allocator

Many programs that have sought to implement some form of learning have involved a teacher. In the very simplest form, a teacher is just an input to a system or part of a system, that indicates when learning should take place.

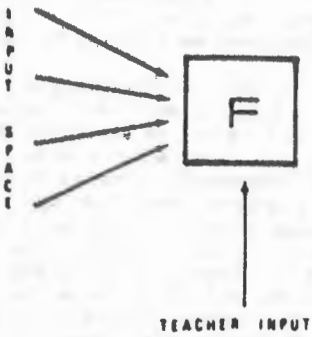


Figure 1
Providing the teacher input is the real problem.

Sometimes, there is an explicit teacher available such as when the Cerebral Motor Cortex teaches the Cerebellum to perform a sequence of elemental movements, here a teacher input directly programs a Purkinje Cell. When the situation is one in which no 'wiser than thou' teacher is directly available (as is probably the case for Cerebral Learning), this model of learning is probably not appropriate. Fahlman's work with NETL has highlighted the point that many issues relating to search can be seen in a different light when processors are not a scarce resource. Search strategies can be viewed as scheduling algorithms, a way of allocating a processor to succesive nodes of a search tree. Teacher inputs can be viewed in a similar light, if the number of processors available for programming is limited, a teacher input is required to allocate one of the available processors for programming. Generating the teacher input is the source of difficulty, it involves knowing what is to be learned, by whom, and when. If processors are not a scarce resource, no teacher input is required for allocation. It is suggested that this is the situation for human Cerebral Cortex. The number of 'programmable elements' in the Cortex, is probably sufficient to learn every brain state encountered in one brains lifetime. This freedom of resource utilisation removes the principle problem of what to learn, for whom, and where. Furthermore, it allows us to concentrate on more important issues of knowledge representation, how to categorize and utilize the learned agents. Far too little is known about Neocortical wiring to persue the neurophysiological analogy presented above, looking at the consequences of such a distributed architecture however has been fruitful.

Structured Knowledge

If processors are in abundance, the problem of allocating processors becomes one of distributing complex knowledge structures across a highly connected network of processors. At the heart of the problem is a strategy for breaking up a complex knowledge structure into component parts that respect its structural nature. A very complex processor could be programmed to represent that knowledge structure. This is the traditional approach. An example of this approach is the production system model. The production system interpreter is the processor and the rules the knowledge structure. One serious problem with this model concerns accounting for the rules. Furthermore, if we are to assume an abundance of processors, it is probably unreasonable to expect the individual processors (plus program) to be that large. The brain's own processors if as numerous as suggested must be very simple in nature. Minsky [7] has highlighted other computational reasons for simple processors.

If a complex item of structured knowledge is to be distributed over a large number of small processors, there arises the problem of how to split the structure and how small the parts should be. A number of observations suggest a solution.

(1) The brain's cortices seem to consist of simple units that compute linear functions of a large number of inputs. These units are constructed from neurons which are threshold devices.

(2) Minsky and Papert [17] showed a number of computational limitations of linear threshold based systems.

(3) Abelson [9] showed how many of the limitations of (2) could be overcome.

(4) Linear threshold functions are known to have serious limitations in terms of the

number of functions that can be computed. As the number of inputs to a LTF increases, the ratio of possible functions of that input space to those that can be computed by a LTF rapidly approaches zero.

There seems to be a contradiction, on the one hand current understanding of cortical physiology suggests that LTF like devices are in great abundance and some interconnection of these devices forms the computational hardware of the brain. These units typically have a very large number of inputs. On the other hand, it appears that there are real limits to what can be computed by a system of such devices (2) and the individual devices are highly restricted in terms of what they can compute, especially for LTF's of many inputs. In a teacher driven system, where processors are a scarce resource this presents a serious problem (much work has been done along these lines with little success). If a processor has a large number of inputs and the teacher wishes to teach a function of those inputs, the chances that the function in question is one that can be computed by a LTF is very small. Making processors an abundant resource turns the situation upon its head. The apparent limitations of LTF's turns out to be their most important property and serves as a source of natural constraint. The functions that cannot be computed by a LTF can be thought of as having a greater structural complexity than can be computed by a LTF. Any complex structure can be decomposed into a structure of LTF's that compute it. This notion leads to an idea that explains an observable psychological phenomenon.

Immediate Learnability

Some things are easier to learn than others. For example, show a child how to open a child proof pill box and he will learn it immediately, no rehearsal is in evidence. Tell a child that the normal body temperature is 98.6°F and he will soon forget it, much rehearsal may be required before retention is achieved. This phenomenon is probably due to a number of interacting mechanisms. One obvious candidate is inteference. The reason of interest to the present discussion concerns the idea of immediate learnability. If an agent can be formed from the programming of a single LTF (implying that all subordinate agents have been learned on a previous occasion), the agent is said to be immediately learnable. When structurally subordinate agents are not present, their formation must preceed the learning of the LTF in question.

Learning by being WRONG

Piaget [15] described learning as `adaptation', an equilibrium between `accomodation' and `assimilation'. This taxonomy helps to distinguish two computationally disjoint forms of learning. Assimilation deals with fitting new information into an already existing structure, whereas accomodation involves adding structure so that new information can be assimilated. The following viewpoint indicates how these processes can proceed bottom up.

Learning by being wrong

The structure to which an event can be assimilated, can be viewed as a prediction about the world. When a prediction turns out to be wrong, a degree of suprise is experienced. Suprise causes predictions about the world to change, and hence change or accomodate new structures.

The notion of suprise is closely related to Winston's [16] idea of Near Miss. Suprise might take many forms. These can be divided into time-related and classificatory suprises. The former concerns predictions of the form 'the event E will occur at time T' being unsatisfied at time T (if T is specifically stated). The latter form of suprise can be implemented with little difficulty, this type of suprise comes from non-temporal prediction. Non-temporal prediction is based on the form of a classification. A new classification is essentially a prediction about events that fall within that category. When subsequent events are later diagnosed as belonging to that category, certain differences between the expectation and the actual event (the auprise) allow the classification to be modified. This form of prediction is time invariant, the emphasis is placed on the classification. Another kind of suprise is the one that comes about by predicting some future event, such as predicting that a loud noise will follow if an object is dropped. If the noise does not ensue, the suprise (unfulfilled expectation) might be used to trigger the learning process. This form of prediction is essentially the same as the notion of recency common in behaviourist theories of learning. The notion of recency is seducingly intuitive, unfortunately it hides many problems, such as a computationally pertinant definition of time. True recency is limited to simple laboratory experiments. This kind of learning is probably an important form of learning, but it seems to defy simple definition and may turn out to be a learned strategy for learning rather than a primitive mechanism. If I predict that the shares of a company X will increase in price substantially in the next few weeks, and buy a number of those shares, I will doubtless be suprised if they do not follow my prediction, and I may learn from it. In the mean time however weeks may have passed, and my suprise may not be instantaneous but spread over a number of days. Non-temporal prediction frees us from the problem of defining time and recency, whilst providing a powerful learning strategy which is entirely data driven. It remains to be seen whether non-temporal prediction is sufficient.

Computational Nature of a Processor

Non-temporal prediction can be computed as local computations on a distributed network of processors. Two issues require detailed specification. First, the nature of the computation performed by the processors of the network, and second, the topology of the proceaor network. We will briefly discuss network topological issues at the end of this paper.

At every node on the network there is a processor which either implements a simple computing agent or has the potential to do so. The computational component of a processor can therefore be broken into two stages, the agent generation process and the agent refinement process. Agent generation will take place whenever an event occurs that has not been categorised previously. Locally, the response to an event will be monitored and used to trigger the agent formation process. This process is more involved than it at first appears, a detailed account of this mechanism can be found in Robertson [13]. Agent formation itself consists of copying the inputs to the processor at generation time and setting a threshold such that if the same event were to happen again, the processor would be activated. However, a strictly identical event will likely never occur again. A concept similarity metric defined as the number of differing inputs is utilized so as to define a category of similar events.

A new agent defines a category based on a sample of one. This acts as a ball park pointer into the landscape. The refinement is best described in terms of hill climbing in the probabiliatic landscape.

The inputs to a processor consist of events within sub-categories identified by other agents. The inputs define an input event space Ψ. Each event $e \in \Psi$ occurs with probability $p(e)$. If the metric space of events is imagined to be spread out on a plane and the probabilities of the occurence of each event on that plane is denoted as a 'height', the common events will appear as mountains. A new agent therefore is a prediction of the form 'the category that I define corresponds to a mountain in the probabolistic landscape'. The refinement process consists of determining the truth of the above assertion, and if true accurately delimiting the mountain. If there is just one mountain in the neighborhood defined by the category, this can be achieved monotonically by taking the arithmetic mean of the subsequent events that fall within the category (cause the threshold of the LTF to be reached). If the landscape contains two or more mountains, the averaging mechanism will result in the threshold being reduced to a point where the processor is available for re-use, because there is no way of representing the situation with single LTF's, first a LTF must be formed to discriminate the mountains, then LTF's can be formed for each individual mountain. The mathematical details of the refinement can be found in Robertson [13].

To summarize, each processor is upon formation placed in a linear programming situation, subsequent events define the landscape. The processor attempts to find a linear prediction function for the landscape. The restriction to a linear function has some computationally powerful consequences as follows.

(1) A Processor attempting to find a linear function of a higher order landscape will ultimately fail and therefore be available for use elsewhere (processors will not be wasted).

(2) Linear functions can be combined to produce functions of higher orders. A high order landscape will naturally decompose into a hierarchy of linear functions. This heirarchy corresponds to the structure of the landscape. A given landscape may have many equivalent structures.

(3) Given a sequence of events and a categorizing algorithm such as the one devised by Marr [14], the structure of the landscape will cause the processes to organize themselves into the appropriate structure without the need for intervention by a teacher.

(4) Linear Threshold Functions (LTF) have a powerful constraining property. A LTF of 2 inputs can implement 14 of the 16 possible logical functions. A three input LTF can implement only 104 of the possible 256. As the number of inputs increases the ratio of functions that can be implemented to the total that are possible quickly approaches zero. In the Mammalian brain it is common to talk of processing elements that have many thousands of inputs. If a linear function is discovered within the landscape presented by the inputs it is probabilistically very unlikely that the linearity was an accident. Restricting Processors to finding and computing Linear Threshold Functions therefore enables the natural structure of the problem to be extracted bottom up, and assuming that many inputs are employed it is highly probable that when a LTF is found, it is a significant one (only 'interesting' agents are learned).

When dealing with highly interconnected systems such as the one proposed here, there arises a problem of connectivity referred to by Minsky [8] as the crossbar problem and by Fahlman [4] as the N^2 problem. The problem is not encountered in Von-Neuman based systems because an address can effectively connect any node to any other node. This use of addresses reaults in a serial based architecture and thereby illiminates any gains that might have been sought with parallelism.

It is doubtfull that any brain would seek to implement such high connectivity. The N^2 problem cannot be ignored, Minsky and Papert [17] made the point

> "- we do not see that any good can come of experiments which pay no attention to limiting factors that will assert themselves as soon as the small model is scaled up to a usable size."

If connectivity is to be reduced, certain assumptions need to be made. A number of

constraining factors have been isolated [12 13] that reduce the growth of connectivity to a linear function of the system size. This can be achieved by making the following briefly stated assumptions about communication.

Assumption - Principle of Locality

Most communication between processors takes place between processors that are physically close. To implement these communication channels, direct connections are allowed for the 'local' communications.

Definition - of Locality

Locality is defined by the product space of two metric spaces. The first is the metric space of 'processedness'. The 'processedness' of a processor defines a partial ordering and is computed as the shortest distance from the processor to an unprocessed input. It is thought to be the exception rather than the rule that a processor that is processing highly processed information will have cause to communicate with a processor dealing with very primitive information. For example, it seems absurd that a classification would involve both highly processed information, such as the recognition of a persons face and low-processed data such as recognition of a hair movement in the nose.

It is contended that this generality is excessive and that this can form the basis for architectural constraint. The second metric is the metric of adjacency. This metric defines a locality for processors within the same processedness band. Processors involved in similar types of work are placed in close proximity. It is thought unlikely that a high degree of communication will take place between processors of highly differing function.

Adjacency is at first less intuitive than processedness, as a source of architectural constraint. An example will clarify the point. Stimulation of a hair on the forearm, might be used as evidence, along with the excitation monitored on a nearby hair to predict the presence of some object touching that region of the arm. Two similar hairs on opposite arms will likely not be considered as evidence in favour of any such hypothesis.

Overlap of neighborhoods

The communication neighborhoods defined by a processors locality overlap one another by a fixed ratio of their size. This overlap is defined by a window and overlap function.

Softening the Restrictions

The above restrictions provide a means of controlling the connectivity problem. Although the restrictions are intuitively valid, if they cannot be defeated in exceptional circumstances they are surely counter intuitive. Rather than saying that all associations occur within a given neighborhood, it would be more acceptable intuitively to present our intuitions about association locality in terms of a distribution such as the Gauss. Although we are unlikely to ever confirm a particular distribution, by stating our intuitions in this way, we do not prohibit any particular association, rather we impose a restriction on the distribution of non-local associations.

The architecture presented can be extended to incorporate non-local connections as follows. If a processor is used to 'relay' a distant connection instead of implementing an agent, distant connections can be achieved. With this scheme, non-local connections have a cost in processors proportional to the distance of the connection (the constant of proportionality is an inverse function of the locality metrics). If a processor is used to implement a 'relay', that processor may not be used to implement an agent. In order that the implementation of relays not intefere with the agent formation process, it is necessary to provide enough topologically equivalent (with the same connections) copies of the processor to allow both 'relays' and an agent to be implemented at every point in the architecture. Replacing every processor (in the original architecture) with a constant number of 'cloned' processors increases the processor connectivity and the total number of processors by a constant factor. If we assume a Gauss distribution of connectivity, the number of 'relay' processors required at each point is σ^2/dl where dl is the diameter of the neighborhood. If $dl = \sigma^2$, the number of processors required at each point specifically to implement relays is just one. In other words, if the neighborhood contains the spread of the distribution, the processors required to implement the relays can be kept as low as one relay processor per agent processor. Each point on the architecture therefore can be viewed as follows.

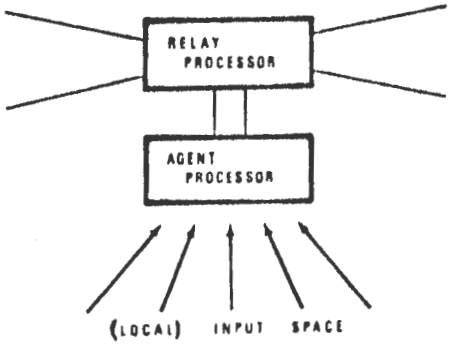


Figure 2

One processor is concerned only with supporting non-local connections, the other only

with non-temporal prediction. Since the two ideas are substantially different, it is natural that the two processors be different in nature. The problem of forming relays, is dealt with elsewhere [18], we can however immediately identify one important difference. The mathematics for non-temporal prediction are based on forming agents with a large number of friends, relays on the other hand will be much more restricted because their function is one of transmitting the prognosis of an agent rather than integration of prognoses. Finally, even the topology of the relay processor can be optimised to take advantage of the fact that the purpose of a relay processor is to communicate with non-local processors (cf. agent processors whose purpose is to integrate information, the bulk of which originates within the neighborhood).

Conclusion

The study of learning, particularly the learning of skills begs many questions. Despite voluminous data from psychologists, theories of skill aquisition remain anecdotal, computationally naive or lacking a suitable computational framework. Recent advances in low level vision have lead to a greater understanding of the computational issues for vision. Although learning lacks the peripheral nature of early vision, the methodology might still be helpful, that is to explore what can be learned knowledge free. At least one part of learning, can be dealt with seperately and the computational problems be solved. Non-temporal prediction may be a small part of learning, time related issues still require explanation. It is believed that this may be easier to do when a greater understanding of the sort of distributed architecture described in this paper has been achieved.

The need to implement the ideas of non-temporal prediction, require considerable architectural constraint of processor connectivity. Computational intuitions based upon the notions of 'processedness' and 'adjecency' allow the crossbar problem to be effectively tamed. Furthermore, strict adherence to pre-defined neighborhoods can be defeated by a supplementary mechanism that provides for a small number of non-local inputs.

Maps of Cortical projections indicate that inter-cortical wiring respects adjacency. It is also interesting that although inspired by different reasons, the metric of processedness discussed in this paper bears remarkable similarity to the level band principle of Minsky [8]. The overlap of locality neighborhoods based upon neighborhood cardinality is related to the 'interfaces' of Abelson [9].

References

1. Marr,D. Palm,C. Poggio, T.
Analysis of a cooperative stereo algorithm
M.I.T A.I. Lab A.I. Memo 446 Oct 1977
2. Marr,D.
Analyzing Natural Images - A computational theory of texture vision
M.I.T. A.I. Lab A.I. Memo 334 June 1975
3. Marr,D.
Analysis of occluding contour
M.I.T. A.I. Lab A.I. Memo 372 Oct 1976
4. Fahlman,S.E.
NETL: A System for Representing and Using Real-World Knowledge
The M.I.T. Press. Cambridge Massachusetts 1979. ISBN 0-262-0609-8
5. Hinton,G
A Parallel Computation That Assigns Canonical Object-Based Frames of Reference
Proceedings of IJCAI-7 1981.
6. Marr,D.
A theory of Cerebeller cortex
J.Physiol 202 pp.437-470 1969
7. Minsky,M.
Plain talk about Neurodevelopmental Epistemology
Proceedings IJCAI-5 June 1977
8. Minsky,M.
K-lines: A Theory of Memory
M.I.T. A.I. Lab A.I. Memo 516 June 1979
9. Abelson,H.
Towards Understanding the Structural Complexity of Computational Problems
10. Waltz,D.L.
A Parallel Model for Low-Level Vision
Coordinated Science Laboratory, University of Illinois
11. Robertson,P.
Computational Issues for Human Memory
University of Essex,England CSCM-3 Nov 1980
12. Robertson,P.
A Computational Study of Learning and Memory
University of Essex PhD Thesis.
13. Robertson,P.
Process Dependant Localized Memory
In preperation.
14. Marr,D.
A Theory for Cerebral Neocortex
Proc. Roy. soc. Lond B 176 pp161-234 1970.
15. Piaget,J.
The origins of intelligence in children
Trans M. Cook Ney York International Universities Press 1952
16. Winston,P.H.
Learning structural descriptions from examples
in The Psychology of computer vision.
Winston, P.H. ed. McGraw-Hill book company NY 1975.
17. Minsky,M. & Papert,S.
Perceptrons - an introduction to computational geometry
The M.I.T. Press. Cambridge MA. ISBN 0 262 13043 2.
18. Robertson,P.
A distributed processing solution to a minimal network problem
University of Texas at Dallas Technical Report.

STATE-SPACE LEARNING SYSTEMS USING REGIONALIZED PENETRANCE

Larry A. Rendell

C.I.S. Dept., University of Guelph, Guelph, Ontario

A new basis is presented for heuristic learning in state-space problem solving. Given a problem and a set of user-defined features, the method first attempts to solve user-selected problem instances. From the results, performance statistics are computed in localized volumes of the feature space. These data allow least squares fitting, and an evaluation function results, for use in future solution attempts. Over repeated iterations of this three-step process: solving, feature space analysis, and statistical regression, the evaluation function improves.

In experiments with the fifteen puzzle, an evaluation function repeatedly resulted which had locally optimal parameters and which consistently solved the puzzle, a unique result. The method is general and has promising extensions.

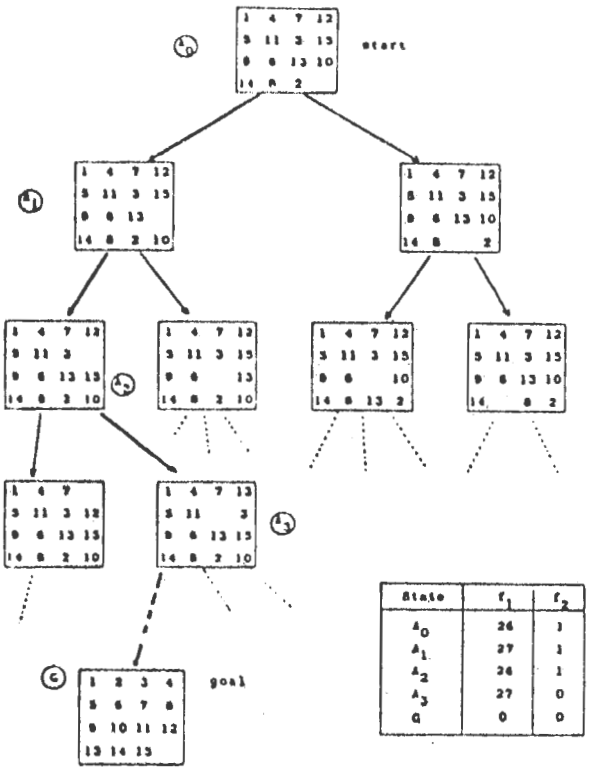
1. INTRODUCTION

Like much A.I. work, this research can be viewed in two basic ways. One is that some aspect of intelligence is being modelled and the chosen problem domain is primarily a rigorous arena in which to test the quality of the "higher level" theory of learning. Rendell (1981) to some extent considers the present method as a partial model of perception. The other viewpoint is that a clearly mechanizable but computationally complex problem exists, and the pragmatic aim is to discover a good heuristic to speed solution, or more ambitiously, to automate development of heuristics, possibly for a class of problems. This is the perspective taken here.

A state-space problem is one that can be formulated in terms of explicitly describable, distinct configurations or states. One state produces others when moves or operators of the problem are applied. For our purposes, we consider a state either to be completely developed by having had all eligible operators applied, or else to be entirely undeveloped. (Partial development is examined in Michie & Ross, 1970, and Mitchell et al, 1981.) A problem instance is specified by a given starting state and (here a single) goal state. In computer implementations, the starting state becomes the root of a search tree, in which operators are edges, and the eventual solution (if obtained) is a path from the start node to the goal. See Nilsson (1971, 1980).

One state-space problem is the fifteen puzzle, a four-by-four block containing fifteen labeled square tiles and a space into which an adjacent tile can be slid (see figure 1). This board puzzle has been the subject of experiment by Doran & Michie (1966), Pohl (1969), Michie & Ross (1970), and Chandra (1972). Schofield (1967) gave a complete mathematical analysis of the simpler eight puzzle, which Gaschnig (1979) also studied in his clear and extensive analysis of heuristics. The fifteen puzzle is a frequently chosen representative of state-space problems because, on the one hand, it is difficult to solve by computer, having about ten trillion states, and on the other hand the problem is easy to mechanize and easily admits useful heuristics, having considerable "structure" or symmetries (see Goldin & Luger, 1975). Although most discussion in this paper involves the fifteen puzzle, the learning system to be described is problem-independent.

In theory, the entire state-space of a problem exists implicitly, but in practice, a search tree is grown explicitly, subject to computer resource constraints. If nodes are developed in the order of their generation, the resulting breadth-first search is impractical for interesting problems, since nodes increase exponentially with tree depth. Hence, guided search is necessary to lead more directly to the goal, wasting fewer nodes and less time (see figure 2).



State	f_1	f_2
A_0	26	1
A_1	27	1
A_2	26	1
A_3	27	0
G	0	0

Figure 1. Fifteen puzzle illustration. A move corresponds to an edge in the search tree. With breadth-first search, a solution (heavy lines) is eventually obtained, but in order to create a heuristic, features can be defined, such as f_1 (distance score) and f_2 (reversal score). In the f_1, f_2 feature space, points close to the origin are better.

To measure quality of a completed search, Doran & Michie (1966) defined the penetrance to be the length of the solution divided by the number of developed nodes. Figure 2 shows an example. A novel refinement of this statistic forms the central concept in the present learning system; penetrance measures are converted into a heuristic to control search.

One general approach to guided search uses the evaluation function to order and select states for best-first development. For example, a simple evaluation function for the fifteen puzzle is the distance score , the sum, ignoring intervening tiles, of the distances of each tile from its home position (see figure 1).

The distance score is an example of an evaluation function H that estimates path length remaining to the goal. More generally, a function can be chosen of the form F = G + H or F = (1-w)G + wH where G(A) represents the estimated shortest length from the starting state to node A; H is the heuristic component, the estimated length from A to the goal; and $0 \leq w \leq 1$. The dependence of search performance on the accuracy of H has been shown in the interesting work of Hart et al (1968), Pohl (1970), Gaschnig (1979), Huyn et al (1980), and Pearl (1980).

In practice, however, it is usually difficult to discover an evaluation function sufficiently accurate for satisfactory performance. For example, while the distance score is an obvious choice for the fifteen puzzle, it often fails because tiles tend to become lodged only close to their correct positions. Doran & Michie (1966) added the reversal score , the number of instances of pairs of tiles being correct except interchanged. In general, several elementary functions or features are typically merged into a heuristic function. But the difficulty is in knowing just how to combine them. To reduce problems of stability, linearity is frequently imposed, so that if $\underline{f}$ is a (column) vector of features and $\underline{b}$ a (row) vector of parameters, the heuristic function is $H = \underline{b}.\underline{f}$ (the vector inner product). Even with this simplification, optimization of $\underline{b}$ is not straightforward since performance measures are required, and, within computer resource constraints, often no solutions whatever can be found to random problem instances, whereas easy ones may not be representative. Despite this, several approaches to parameter adjustment have been quite successful, among them Samuel (1963, 1967), Michie & Ross (1970) and Berliner (1979). While these methods allow learning during a single search, the present system utilizes only completed searches, beginning with easy problem instances and improving its heuristic function iteratively.

One way of facilitating heuristic learning is to define the evaluation function in terms of correlations between observed solution participation

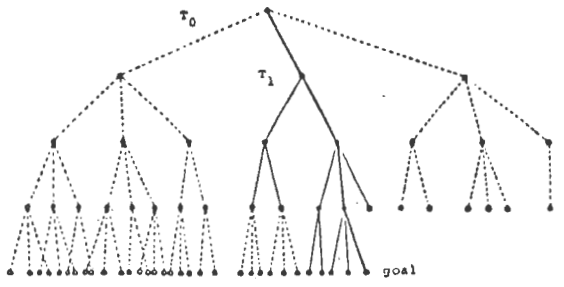


Figure 2. Two superimposed searches for a short solution. T_0 is breadth-first and T_1 (solid edges only) is more direct. The Doran & Michie penetrances: $P(T_0) = 4/24 = 0.17$, whereas $P(T_1) = 4/6 = 0.67$.

of states and their distinguishing characteristics (c.f. Slagle & Bursky, 1968, Slagle & Farrel, 1971, and Simon & Kadane, 1975). In particular, the present method regionalizes penetrance in the user-defined feature space formed from the components of the evaluation function. More precisely, if r is a volume of the feature space and T is a search tree, then the penetrance p(r,T) is the number of solution nodes from T which map into r, divided by the total number of developed nodes falling within r. A simple example is shown in figure 3. As described later, it is these localized penetrance values, that, after some manipulation, provide state evaluation. So, rather than estimate path distance, our evaluation functions rank states in a range of probabilities [0,1]. The mediating feature space allows state discrimination and heuristic modification. Probabilistic performance measures and feature space differentiation are topics of control theory and pattern recognition; see Fu (1978), Nananabe (1972), Hunt (1975) and Sklansky & Wassel (1981).

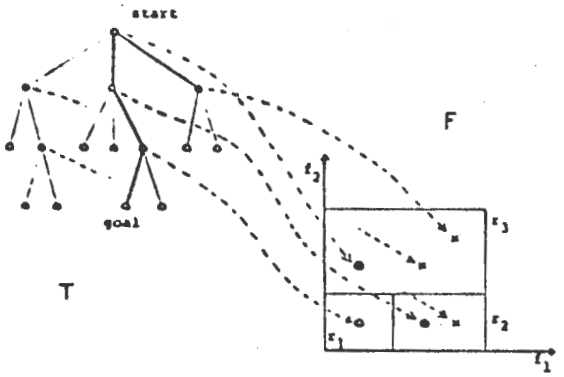


Figure 3. Localizing penetrance. Shown are a very small search tree T and feature space F into which developed nodes are mapped. The full space penetrance of T is 3/6 = 0.5, whereas localization in F gives (e.g.) three differentiated values: $p(r_1,T) = 1/1 = 1.0$, $p(r_2,T) = 1/2 = 0.5$, and $p(r_3,T) = 1/3 = 0.3$. Such solution participation proportions can be used to generate a probabilistic evaluation function.

In the present penetrance learning system PLS1, the state, feature and ranking spaces interact in an iterative method which employs three steps: (1) solving, (2) manipulation of resulting statistics in the feature space, and (3) parameter computation for the penetrance estimating heuristic function to be used for state evaluation in the next round of solving. More specifically, at the outset the user defines a feature vector $\underline{f}$. Then, at each iteration t: (1) The solver accepts a user-selected set of problem instances and attempts solution, guided by the penetrance predicting function H_{t-1}. Initially, H_0 gives breadth-first search. (2) Assuming at least one successful search, penetrance measures are localized or clustered in the feature space, where they are combined with previous values for normalization and refinement. This step is complex; details are given later. (3) The penetrance statistics are regressed on feature values for least-squares determination of the parameter vector $\underline{b}_t$ to give the evaluation function $H_t = \exp[\underline{b}_t \cdot \underline{f}]$. This heuristic estimates the probability of solution participation. Figure 4 shows an example. In their interesting work, Mitchell et al (1981) use a different sort of iterative refinement, along a general-to-specific partial ordering in a "version space" of incompletely learned heuristics.

In the terminology of Buchanan et al (1978), the performance element of PLS1 is the solving step (1) T ($\underline{b}$), an algorithm whose standardized control structure is the parameter vector $\underline{b}$. The higher level learning element comprises the second two steps (2) feature space penetrance manipulation, and (3) regression. The learning element of PLS1 improves $\underline{b}$ in an unsupervised manner, indirectly through the feature space. Along with $\underline{b}$, feature space penetrance measures constitute the blackboard of structures for communication between the performance and learning elements. Interestingly, although local optima have been located, there is no critic directly examining the quality of a parameter vector.

Parameters in an evaluation function for the fifteen puzzle have not previously been optimized in any way, and only one other program has consistently solved the puzzle. Chandra (1972) wrote a successful bidirectional search program that used problem reduction, placing outer gnomons first; this was not a learning system. The present PLS1 has repeatedly generated an evaluation function with locally optimal parameters which solves all of a set of 50 random puzzle instances.

Generality, stability, and extensions of PLS1 are discussed after a closer examination of the system.

2. BASIC CONCEPTS AND STRUCTURES

Forming the basis for PLS1 is the solution participation proportion, the localized penetrance. This section defines an idealized normalization of this statistic and discusses the "region", a convenient notation for associating penetrance values with feature space volumes.

Given a feature vector $\underline{f}$ and set of search trees T, define the total count $t(r,T)$ to be the number of developed states from trees in T which fall into volume r in the feature space determined by $\underline{f}$. Define the good count $g(r,T)$ similarly, except each node counted must also participate in a solution in a search tree in T. The elementary penetrance $p(r,T)$ of r for T is $g(r,T)/t(r,T)$. It is precisely the conditional probability that a state $A \in T \in T$ is in the solution in T, given that $\underline{f}(A) \in r$. Figure 3 shows a simple example.

As figure 2 suggests, the elementary penetrance is affected drastically by the heuristic used in the search. Since they result in fewer wasted nodes, good evaluation functions bias elementary penetrance values upwards. Moreover, the difficulty of the problem instances solved may also influence elementary penetrance. For example, in a uniform breadth-first tree, if B is the branching factor and d is the depth at which the goal is found, the number of nodes developed is roughly B^d, so the full feature space penetrance is about d/B^d. Actually, this effect of lower penetrance with harder problem instances does not generalize when penetrance is localized. A "good" feature space volume tends to have high penetrance, independent of poorer volumes, where the wasted nodes congregate. See figure 4. Nevertheless, a more subtle phenomenon does remain: The importance of some features may depend on the difficulty of the problem instance. For example, in the fifteen puzzle, reversals cannot occur in very easy starting states. This is reflected in figure 4, where the reversal score is found to discriminate only in later iterations when harder instances can be solved.

Since we want complete and accurate information in order to create the best evaluation function, we could imagine the ideal case in which all possible instances of a problem were solved breadth-first, to give the exhaustive search tree set T_a. This would be a perfect collection of unbiased shortest solution searches. The result would be $p(r,T_a) = \bar{p}(r)$, the true penetrance of volume r, an absolute measure of the worth of a feature space area r. Approaching this ideal in practice, however, presents considerable difficulty. The dilemma is that search trees for typical, random problem instances are simply too large to generate breadth-first, and while enough representative problem instances might be solved using some heuristic (if a fairly good one is already known), the elementary penetrance measures from the resulting search trees are badly biased in an unknown way.

But, in PLS1, the true penetrance $\bar{p}$ is first approximated, then the estimates are improved iteratively. Initially, breadth-first search trees are obtained for easy (solvable) problem instances, then a heuristic derived from this foothold is exploited to solve harder problem instances. The consequent elementary penetrance values, though biased if untreated, are normalized to estimate true penetrance before contributing to the accumulating heuristic base.

As a useful notation for discussion, and as a blackboard structure for system communication and manipulation, we provide a means for associating penetrance with its feature space location. A region $R = (r, u, e)$ is a feature space volume r, real number $u \in [0,1]$, which is interpreted as a probability, and real $e \geq 1$, which represents an error factor estimate for u. The value u can be an elementary penetrance, in which case R is an elementary region, or u can estimate the true penetrance $\bar{p}(r)$, in which case R is a (true penetrance) estimating region. If r contains just one point, R is a point region.

In a PLS, sets of estimating regions $\{(r, \bar{p}, e)\}$ house the entire heuristic essence. The evaluation funotion H computed from such a set is given by $\log H = \underline{b}.\underline{f}$ where $\underline{f}$ is the feature vector and $\underline{b}$ is the parameter vector found by error-weighted regression. (Reasons for the logarithmic form and other details are discussed in Rendell, 1981). More recent PLS's extend the region definition to $R = (r,u,e,\underline{b})$ to provide non-linear evaluation.

3. PLS1 HIGHLIGHTS

This section mentions some of the choices made in implementing PLS1. Details and further examination are given in Rendell (1981, 1982).

Although practically unattainable, the most desirable heuristic information would be in the primary form of an exhaustive set $R = \{(r, \bar{p}(r), 1)\}$ of perfectly accurate true penetrance estimating point regions where each r contains just one point and R covers the whole feature space. Even so, Berliner's (1979) results concerning the detrimental effect of local feature space "blemishes" suggest these data should be smoothed. In addition to this consideration of evaluation stability, when

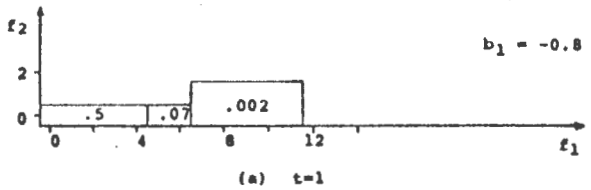
iterative improvement was attempted, Rendell (1981) found problems of anamalous "drift" away from useful heuristic function convergence without some control. Hence the (log-) linear model $\log H = \underline{b}.\underline{f}$ is fitted, where $\underline{b}$ and $\underline{f}$ represent the parameter and feature vectors, and H is the predicted true penetrance. A stepwise regression algorithm is used, which rejects less general and useless features.

The data used for this statistical regression estimate true penetrance. In the first iteration, easy problem instances are solved breath-first to give a set of search trees T. Each estimating region $(r, p(r,T), e)$ is constructed assuming the elementary penetrance $p(r,T)$ represents the true value. In subsequent iterations, the true penetrance estimate β is obtained for each updated estimating region (r, β, e) by a process comparing former data with current elementary regions, a bootstraping approach to penetrance normalization. For matching feature space volumes r, penetrance values of new estimating regions and former estimating regions are combined in an error-weighted average to give a revised true penetrance estimate. Heuristic information collects and alters in cumulative regions.

The elementary penetrance observations are taken for point regions but the normalized true penetrance estimates are clustered into sizable volumes of the feature space, housed in non-point cumulative regions. A set of these cumulative estimating regions, the primary blackboard structure, partitions the feature space (see figure 4). The regions are rectangular and aligned with the axes, reducing the complexity of the clustering algorithm that determines the partition (and apparently not vitiating the system). This information compression relieves storage and facilities PLS extensions (discussed in section 4).

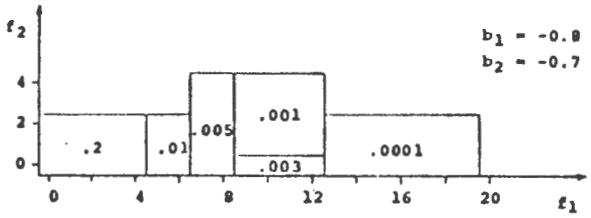
In the first system iteration, the feature space is undifferentiated until this clustering is performed (typically giving about three rectangles). In any later iteration, each cumulative region becomes an input for the clustering algorithm, as the established partition gradually becomes more refined (typically about twenty rectangles after half a dozen complete system iterations, solving, clustering, regressing). The clustering algorithm is efficient; typically ten features are handled using less time than the solving step.

In all, a PLS1 iteration t has three steps: (1) solving step: the search tree set $T_t = T(H_{t-1}, P_t)$ is obtained from a selected problem instance set P_t using heuristic evaluation function H_{t-1} (trivial in the breadth-first initial iteration); (2) clustering step: the cumulative region set $C_t = C(C_{t-1}, T_t) = \{(r, \beta, e)\}$ is either formed (for t=1 when C_0 is a single all-embracing region with undefined penetrance) or else modified (for t>1, by penetrance revision and partition refinement); and (3) regression step: the improved evaluation function $H_t = H(C_t)$ is computed by (error-weighted) stepwise regression of penetrance β on centers of the feature space rectangles r. Figure 4 illustrates. (The appendix shows state evaluation.)



(a) t=1

Figure 4. Typical early progress for the fifteen puzzle and usual illustrative features. Shown are cumulative regions with true penetrance estimates within (output from step 2) and non-zero parameters (step 3 result), after (a) iteration 1 and (b) iteration 2.



(b) t=2

4. CONCLUSIONS AND EXTENSIONS

In experiments with the fifteen puzzle, parameters, for six features similar to the two in examples here, repeatedly converged to a local optimum in the parameter space, resulting in a linear evaluation function which solved all of a set of 50 random puzzle instances. Detailed results for a formula manipulation problem and for the fifteen puzzle are given in Rendell (1981), and highlights are summarized in Rendell (1982). This section discusses more general issues.

Although the local optimum found was not far from the parameter vector at just one or two iterations (final parameters within a factor of about 2 or 3 of early values), several facts stand out. First, the discovery did not use any

overall performance measure as feedback (no optimization objective function). Also, alternate methods of finding the exact optimum even after approximate location were less efficient, perhaps partly because PLS1 utilizes as many quantities as there are regions, rather than just one datum per solution search. Finally, true penetrance is valuable.

This penetrance learning is applicable when solutions proceed roughly along a low to high true penetrance path in the feature space. If, instead, for a given state-space problem, localized penetrance and feature space solution paths were highly varied among different problem instances, statistics for "average" searches would frequently be misleading. This phenomenon could be called solution irregularity. An example is shown in figure 5(a). One might suppose that solution irregularity is one way of characterizing problems not amenable to a PLS-like method, problems which need "planning". But as discussed in different contexts in Watanabe (1972, pp. 563-4) and Feldman & Yakimovsky (1974), other paradigms are theoretically equivalent to a feature space approach. In the latter, successful implementation relies on appropriate feature abstraction and suitable combination of the features into a good heuristic. Solution irregularity is dissolved by adding further discriminating features; see figure 5(b).

However, assuming enough relevant and easily computable features can be realized, the problem of how to merge them still remains. Among the limitations of PLS1, perhaps the most serious is that it attains stability through linearity in feature combination. In preliminary experiments with features having suspected strong interactions, a good evaluation function did not result.

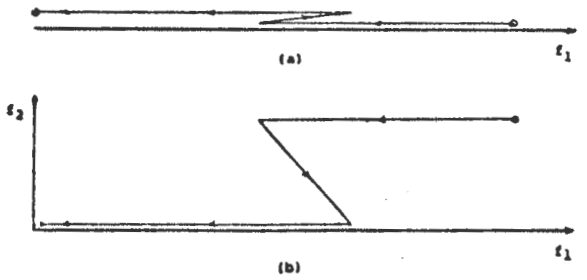


Figure 5. Enough good features discriminate solutions. Irregular solution paths traced in feature space become unsurprising when dimensionality is increased. Again f_1 and f_2 are the distance and reversal scores. In the one-dimensional f_1 space (a), solutions sometimes follow strange paths. However, they appear normal in the f_1, f_2 space (b), where the distance score is temporarily worsened to unclog a reversal.

Other penetrance-based systems, extensions of PLS1 designed for both stability and power, seem promising. In one, feature nonlinearities are accommodated through multiple, piecewise linear models $H^{(i)}$, one for each cumulative region R_i. At system iteration t, suppose there are m_t cumulative regions, therefore m_t evaluation functions $H_t^{(i)} = \exp(\underline{b}_t^{(i)}.\underline{f})$ $(1 \le i \le m_t)$. Heuristic $H_t^{(i)}$ is obtained through stepwise regression as in PLS1, but now the contribution from each region R_j is weighted by a value $1/d_{ij}^2$, where d_{ij} represents a "distance" between the principal region R_i and the contributing one R_j, the idea being that regions in the neighbourhood of R_i more accurately reflect penetrance behavior within R_i. The most obvious choice is the Euclidean feature space distance $d_{ij} = \sqrt{(\underline{c}_i - \underline{c}_j).(\underline{c}_i - \underline{c}_j)} + \epsilon$ where $\underline{c}_i$ and $\underline{c}_j$ are the feature space centers of the regions (and ϵ is a small number to avoid a zero for $i = j$). However, since some features are more important than others, and in fact some are irrelevant, this d_{ij} would be misleading. A better choice might be to multiply each feature value by its parameter, converting d_{ij} into a penetrance-related measure, but it is the parameter vector itself that we want to determine. This suggests another iterative algorithm to improve the estimate of $\underline{b}_t^{(i)}$ in successive regressions. Assuming convergence in $k_t^{(i)}$ repetitions, the total number of stepwise regressions in (a primary) system iteration t is $\sum_{i=1}^{m_t} k_t^{(i)}$. This multiplicity probably would not slow the system significantly, since in PLS1 the regression step is at least two orders of magnitude faster than the solving step. This scheme modelling feature interactions is completed by a similar distance-weighted state evaluation in the solving step.

Another extension, compatible with this one, has additional benefits. It is based on Holland's (1975) genetic model which is capable of locating absolute optima. The natural example of a structure is DNA, within which one of a small number of alternatives (alleles) occurs at each location. In any generation, there are many individuals, each characterized by a distinct structure. Certain genetic operators (e.g. crossover, inversion) act on structures to produce offspring. Some fitness function is used stochastically to select structures for procreation according to their survivability. As Holland shows in his substantial book, genetic plans

exploit inherent advantages of parallelism, testing schemata, whole sets of structures in a single trial, and the resulting gain is much greater than suggested by the simple multiplicity of structures.

In a proposed genetic PLS, the structure is the cumulative region set. At each iteration or generation t, there are q_t solving steps, each attempting to solve the same problem instances. In the normal PLS1 manner, each of these leads to q_t region handling steps which independently produce q_t refined cumulative region *sets* $C_t^{(k)}$ $(1 \leq k \leq q_t)$. These are pooled and their members are judged. The fitness of a region R is $\mu[R]$ = (average number of states developed in all q_t solution trials of the problem instance training set) / (average number of states developed in solutions associated with region sets $C_t^{(k)}$ $(1 \leq k \leq q_t)$ containing a region "like" R). Using present PLS1 mechanisms, it is straightforward to define "like" regions, and beyond the multiplicity of trials, the global measure μ does not worsen complexity since it is a by-product of PLS1 penetrance refinement.

From $\bigcup_{k=1}^{q_t} C_t^{(k)}$, regions are chosen according to their fitness, by a stochastic selection mechanism which includes a tendency to cover the whole feature space. The chosen regions are collected and their centers perhaps reclustered before evolving into one of several new structures $C_{t+1}^{(k)}$ $(1 \leq k \leq q_{t+1})$. Each of these q_{t+1} region sets determines a different evaluation function for the following q_{t+1} parallel solving steps.

Although it is several times more expensive to compute than PLS1, this genetic plan has important advantages. The μ-weighted trial allocation gradually increases the proportion of good heuristic representatives (successful regions), allowing shifts as the environment changes (as problem instances become harder). One would also expect good stability, even with small q_t.

BIBLIOGRAPHY

Berliner, H. (1979): "On the Construction of Evaluation Functions for Large Domains," Proc. Sixth International Joint Conference on Artificial Intelligence, 1979, 53-55.

Buchanan, B.G., Johnson, C.R., Mitchell, T.M., & Smith, R.G. (1978): "Models of Learning Systems," in Belzer, J. (ed.), Encylopedia of Computer Science and Technology, 1978.

Chandra, A.K. (1972): Stanford University fifteen puzzle "gnomon" program.

Doran, J. & Michie, D. (1966): "Experiments with the Graph Traverser Program," Proc. Roy. Soc. A, 294 (1966), 235-259.

Feldman, J.A. & Yakimovsky, Y. (1974): "Decision Theory and Artificial Intelligence: I. A Semantics-Based Region Analyzer," Artificial Intelligence 5 (1974) 349-371.

Fu, K.S. (1978), "Pattern Recognition: Discriminant and Syntactic Methods," in Belzer, J. (ed.) Encyclopedia of Computer Science and Technology 12 , 1978, 28-49.

Gaschnig, J.G. (1979): Performance Measurement and Analysis of Certain Search Algorithms, Dept. of Computer Science CS-79-124 (PhD Thesis), Carnegie-Mellon University, 1979.

Goldin, G.A. & Luger, G.F. (1975), "Problem Structure and Problem Solving Behavior", Proc. Fourth International Conference on Artificial Intelligence, 1975, 924-931.

Hart, P. Nilsson, W.J. & Raphael, B. (1968), "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," IEEE Trans. Sys. Sci. and Cyberneetics SSC-4 (1968) 100-107.

Holland, J.H. (1975): Adaptation in Natural and Artificial Systems, University of Michigan Press, 1975.

Hunt, E.B. (1975): Artificial Intelligence, Academic Press, 1975.

Huyn, W., Dechter, R., & Pearl, J. (1980): "Studies in Heuristics, Part 6: Probabilistic Analysis of the complexity of A*", Artificial Intelligence 15 (1980) 241-254.

Michie, D. & Ross, R. (1970): "Experiments with the Adaptive Graph Traverser," in Meltzer, B. and Michie, D. (ed.), Machine Intelligence 5, American Elsevier, 1970, 301-318.

Mitchell, T.M., Utgoff, P.E. Nudel, B., & Banerji, R. (1981): "Learning Problem-Solving Heuristics through Practice," Proc. Seventh International Joint Conference on Artificial Intelligence, 1981, 127-134.

Nilsson, N.J. (1971): Problem Solving Methods in Artificial Intelligence, McGraw-Hill, 1971.

Nilsson, N.J. (1980): Principles of Artificial Intelligence, Tioga, 1980.

Pearl, J. (1980): "The Utility of Precision in Search Heuristics", UCLA-ENG-CSL-8065, University of California, 1980.

Pohl, I. (1969): "Bidirectional and Heuristic Search in Path Problems," Stanford Linear Accelerator Report 104, Stanford, 1969.

Pohl, I. (1970): "First Results on the Effect of Error in Heuristic Search," in Meltzer, B. and Michie, D. (ed.), Machine Intelligence 5, American Elsevier, 1970, 219-236.

Rendell, L.A. (1981): An Adaptive Plan for State-Space Problems, University of Waterloo Research Report CS-81-13 (PhD. Thesis), 1981.

Rendell, L.A. (1982): A New Basis for State-Space Learning Systems and a Successful Implementation," University of Guelph Technical Report CIS82-1, January, 1982.

Samuel, A.L. (1963): "Some Studies in Machine Learning Using the Game of Checkers", in Feigenbaum, E.A. and Feldman, J. (ed.) Computers and Thought, McGraw-Hill, 1963, 71-105.

Samuel, A.L. (1967): "Some Studies in Machine Learning Using the Game of Checkers II--Recent Progress," IBM J. Res. & Develop. 11 (1967) 601-617.

Schofield, P.D.A. (1967): "Complete Solution of the 'Eight Puzzle'," in Collins, N.L. and Michie, D. (ed.), Machine Intelligence 1, American Elsevier, 1967, 125-133.

Simon, H.A. & Kadane, J.B. (1975): "Optimal Problem-Solving Search: All or None Solutions," Artificial Intelligence 6 (1975) 235-247.

Sklansky, J. & Wassel, G.N. (1981): Pattern Classifiers and Trainable Machines, Springer-Verlag, 1981.

Slagle, J.R. & Bursky, P. (1968): "Experiments with a Multipurpose, Theorem-Proving Heuristic Program," JACM 15 (1968) 85-99

Slagle J.R. & Farrel, C. (1971): "Experiments in Automatic Learning for a Multipurpose Heuristic Program," C. ACM 14 (1971) 91-99.

Watanabe, S. (1972): Frontiers of Pattern Recognition, Academic Press, 1972.

APPENDIX—NODE EVALUATION

The table below shows how the five labelled nodes in figure 1 are evaluated by the two penetrance predicting functions of figure 4, $H_1 = \exp(-0.8f_1)$ and $H_2 = \exp(-0.8f_1 - 0.7f_2)$.

State	f_1	f_2	H_1	H_2
A_0	26	1	9.3×10^{-10}	4.6×10^{-10}
A_1	27	1	4.2×10^{-10}	2.1×10^{-10}
A_2	26	1	9.3×10^{-10}	4.6×10^{-10}
A_3	27	0	4.2×10^{-10}	4.2×10^{-10}
G	0	0	1	1

Emotional Robots will be Almost Human

D Julian M Davis
Department of Computer Science
University of Waterloo
Waterloo, Canada N2L 3G1

ABSTRACT

We survey theories of the nature of *emotion* and consider the implications for Artificial Intelligence models of cognition. We argue that for an organism to have *emotions*, it must possess *feelings* in the sense of innate feedback from its physical body (implementation substrate) to its mind and cognitive processes. The models of mind commonly adopted in Artificial Intelligence - Information Processing models - tend to focus on cognitive 'rational' processes to the exclusion of 'feelings'.

1. INTRODUCTION

A recent paper by Sloman and Croucher[1] discusses the idea that eventually some robots or intelligent systems might be said to have emotions. They discuss the concept of Emotion, and relate it particularly to the concept of Motive.

In this paper, we take another look at the concept of Emotion, and briefly survey the recent literature on the topic. We also examine specifically the 'Causal-evaluative' theory of Lyons[2] and others, and the 'Structural' analysis of de Rivera[3].

We shall argue that a theory of *feeling* is essential to an adequate conceptualization of *emotion*, and that this is missing from the Information Processing (IP) models of cognition generally adopted in Artificial Intelligence studies. We offer some suggestions on how *feeling* can be handled in an IP-type analysis of mind.

It may seem somewhat perverse to examine such rarified (high-level) concepts as 'emotion' in the present context of computational studies of intelligence. However, we argue that it is necessary sometimes (perhaps especially in Cognitive Science) to look ahead and grapple with problems which we are not yet ready to solve. This paper is intended in part to suggest directions for future research. There is a marked lack of detailed computational theories in this area, and many questions remain to be studied.

2. THEORIES OF EMOTION

Over the centuries, a wide variety of theories of emotion have been propounded. Our survey will follow Lyons'[2] for the most part, but we shall consider especially what value these theories might have in guiding proposed architectures for minds. In fact there is an enormous literature on Emotion, but not much is immediately relevant to AI.

Sometimes it can be helpful, in analyzing a concept, to examine dictionary definitions, and especially the examples cited for various usages. This helps guard against the danger of focussing too much on only part of the total meaning of a word. However, we must not be mislead by the presuppositions of the lexicographers, either.

My dictionary[4] defines *emotion* thus (abbreviated):

> emotion: n. from: *emovere* to remove, move away; shake, upheave
> 1. stirring up, agitation, excitement of feelings
> 2. any of feelings—contrasted with mental processes of reasoning [joy, grief, hatred, love, fear, etc.]
> (popularly) emotion of heart, contrasted to intellect

2.1. Feeling Theories of Emotion

This dictionary definition links *emotions* closely with *feelings*. That association is well embedded in our culture. 'Feeling' theories of emotion date from Descartes or earlier, and describe emotions as mental events of one sort or another which involve feelings as the essential component. This approach, however, is incomplete.

The definition noted above for 'emotion' leaves only implicit (mainly in its etymology) another aspect, that in many respects an emotion can be something of a *motive* for *action*. A person in the grip of an emotion, such as anger or love, is thereby moved to act in certain ways (or to avoid certain actions). Indeed, sometimes emotions in people are diagnosed by how (and whether) they manifest themselves in behaviour. A stock example is "I didn't realise how angry I was until I found myself pounding on the table."

Feeling theories of emotion do not account for this satisfactorily, since feelings, as such, do not motivate behaviour. Also, if emotions are merely mental events of feeling, there is a mind-body problem in how can they affect behaviour. Emotions may involve feelings, but there is something more.

2.2. Behaviourist Theories

Another group of theories may be called 'Behaviourist'. These attempt to identify emotions by (resulting) behaviour (Jim punched Robert on the nose. Mary's eyes sparkled and her pulse rate increased when Andrew came into the room.) Behaviour is understood to include observable physiological changes (in pulse rate, skin flush, breathing, etc.)

However, careful examination of the facts shows that emotions (in humans) do not possess a one-to-one mapping with sets of observables. The same outward behaviourist manifestations can indicate any of several different emotions, depending on context. (Jim and Robert may have been in a boxing match, or may be acting in the theatre.) People only label physiological changes and overt behaviour as 'emotion' if appropriate cognitions are present.

There is a very considerable literature on emotions from a behaviourist perspective, largely founded on experiments with animals. This research does explore particularly the correlations of outward signs of emotion with physiological aspects such as body chemistry (hormones, etc.) and with activity in the Central nervous system and the Autonomic nervous system[5].

2.3. Psychoanalytical Theories

Psychoanalytical theories, mainly derived from Freudian perspectives, interpret emotions as manifestations of unconscious influences on the ego. The fact that much mental activity is unconscious is now generally accepted. This perspective, also, has a very large literature, mainly in the context of treatment of emotionally disturbed people.

2.4. Jungian Theories

Jungian theories do not have much to say about emotion as such. They make a claim, however, which is somewhat relevant. All cognitions, mental associations, 'complexes', and thoughts are said to be accompanied *intrinsically* by a *'feeling tone'*. This is something of a radical challenge to any theory of mind based (as many are in AI circles) predominantly on the ideas of Information Processing. Jungian theories say that *feelings* in the form of (perhaps unanalysable) attractions and repulsions, likes and dislikes, are part and parcel of *all* mental events. Part, in a sense, of the low-level neural functioning of the brain (though the implementation aspects of this are not defined).

This claim, of the centrality of 'feeling tones' in mental processing, can be correlated with the observations of behaviourist 'physiological psychology', which see feelings and emotions as essentially 'psychic side-effects' of activity in the CNS or ANS

Feelings and Motives

The Jungian theory offers grounds for regarding not only feelings, but also *motives*, and *goals* or *desires* as intrinsic to animal mental functioning. The intimate connection between the mental phenomenon of 'feelings' and the physiological activity in the nervous system provides a direct mechanism why the animal should have purposes; they can arise at least from pure hedonism—maximisation of pleasurable feelings. Although this appears obvious, it cannot be accommodated by the purely cognitive II³ models now in use.

It is a trivial philosophical observation that no amount of *information* ever leads of itself to a value judgement, nor to a motive for action. The mental event that something is *valued*, or that some action or response is called for in a particular situation, must involve another element besides the information inputs. This other element is seen in its most primaeval form in simple reflexes, for instance where a mosquito homes in on sources of warm humid air, having been stimulated to activity by carbon dioxide. Of course, we normally reserve the words 'motive', 'goal' and 'desire' for more elaborate mental processes involving complex information processing and cognitions

2.5 Motive Theories of Emotion

We have remarked that emotions are not just feelings (or just physiological disturbances) but are commonly associated with a spurring towards action, i.e. a *motive*; these words have a common etymological root:

motive: n. cause of movement
1. That which influences desires, incites will in a direction, induces specific action; inner force causing specific actions
2. Predominant feature, idea or theme underlying or running through an artistic composition, etc.

This has led to theories, e.g. Leeper[6,7], which see *emotions* as special kinds of *motives*. Lazarus and Magda Arnold have also argued this, and Sloman and Croucher[1] hold a position close to this. For instance, an angry person is angry *at* someone or some group whom s/he believes has damaged his/her interests, and s/he may well feel an impulse to retaliate. This approach dates back to Aristotle.

When we examine our usage of the word 'emotion', we find yet another element is also involved besides the motivation to action. The motive to retaliate (in the above example) is consequent on a prior *perception* that 'interests have been damaged' (e.g. the ball broke my window, or your spouse spent the money you were saving for a holiday, or the postal workers were on strike again).

2.6. Causal-Evaluative Theories

This refinement of the motive theory leads to a Causal-Evaluative theory of emotion. In this, an emotion is said to consist of a subject *perceiving* or *evaluating* a situation as significant (vague word!). This perception or evaluation involves reference to the subject's expectations and desires. It leads to (causes) some combination of changed physiological responses (in living creatures), feelings, and a motive to react. In this theory, the initial evaluation is central, *and* depends on the subject's purposes and expectations. Furthermore, to qualify as an emotion in normal usage, the evaluation and its consequences (physiological and mental) should be 'automatic' — not directly controlled by the subject's will-power.

To explicate, visual perception is also an 'automatic' process, and for the most part we cannot directly *will* to see or not see an object in the visual field presented to our eyes. Similarly, we cannot *will* to feel or not feel a particular emotion, though of course we can influence our emotions indirectly, such as by leaving the room or consciously redirecting our attention. The Motive component may or may not result in overt action. Fear of an enemy may include a desire to flee, but can result in paralysis of action depending on the circumstances.

However, the emotion may not directly involve a specific *motive* at all. For instance, in the grief of bereavement it is hard to identify any motive as resulting from the sense of loss. (This will be related to the concept of 'fluid' and 'fixed' emotions to be discussed later.)

Evaluations or Perceptions

The evaluation involved in an emotion generally involves prior mental states, expectations and beliefs. For instance, if Emma comes into the room while I am eating lunch, and I display signs of agitation, you would need to know more about my prior mental state to judge what emotion I might be feeling. For instance, I

could be secretly in love with her, or perhaps I dislike her very much. Alternatively, I might have a private message for her which is making me anxious, or I might fear she will embarrass me. Alternatively, I might be getting indigestion, and her arrival was a coincidence.

Note that the *evaluation* leading to an emotion is somewhat similar to those processes normally regarded as 'perception'; they are automatic and present an analysis of the situation. However, while 'perception' generally involves reference to beliefs and expectations [1, 8] it does not involve (as usually understood) reference to our *purposes* and *desires*. The evaluation for an emotion typically *does* involve reference to our purposes and desires. We might infer that feeling an emotion involves a more elaborate mental machinery than is required for perception. Or it may be that the underlying information processing mechanisms are essentially similar, but have access to 'higher level databases' in the case of emotions.

2.7. Cognitive theories

Sloman's[1] analysis is consistent with the causal-evaluative position, though perhaps over-estimating motives as compared with the evaluation. It seems forced to describe a *pain* as a kind of 'motive'. When you sit on a thumb-tack, the reflex actions (a gasp, and upward leap) are hardly mediated by 'motives' as the term is usually understood. The whole process involves only phylogenetically ancient parts of the nervous system (reflexes), and it is a moment later that higher cognitive processes assimilate the signals that your rear end has been punctured. Motives (and possibly emotions) then follow, as in removing the offending object before sitting down again, and maybe feeling anger at the careless (or deliberate?) placement of the tack.

Again, as suggested earlier, it is an unusual usage to describe the pain in the emotion of grief as a 'motive'. And similarly for the emotion of 'longing' (for someone absent). These 'fixed' emotions tend to persist for a period of time, and are not amenable to being resolved by taking any outward action. Also there is no explosion of emotion with cathartic effect in these cases. (Compare the emotion of anger at someone's action, where permitting expression of the emotion goes a long way towards removing it.)

Sloman does mention rightly the intrusive involuntary nature of emotions as mental disturbance, though mental disturbance alone does not compose an emotion (*pace* some behaviourist psychologists). Such disturbances could indeed be suffered by an artificial intelligence. If they are consequent upon an evaluative process as described above, perhaps even a robot *might* be said to be feeling an emotion. However, this is definitely stretching the word a bit in favour of the robot, unless in some sense the robot also has feelings. This is not a matter of physiological *disturbance*, but as suggested earlier is perhaps dependant on whether the cognitive machine directly 'possesses feelings' as a primitive 'datum'. More precise formulations are needed in this area. The word 'feeling' has diverse meanings in English.

feeling

I n. 1. physical sensation, sentience;
power of, capacity for, experiencing sensation.
2. a. specific sensations; b. specific emotion or sentiment; c. a premonition, intuition
3.[pl.] a. the emotions;
b. (specifically) kindly, generous sentiment, etc
4. excitement of mind; esp. angry, etc. sentiment
5. intuitive aesthetic appreciation, delicate and just sensibility (e.g. feeling for beauty)

II adj. having, animated by, expressing, strong feeling and emotion.

Note that it is distinctly anomalous for someone to say "I love (fear, envy, etc.) Janet, but never have any feelings towards her." In fact, a common way to report an emotion is to say, e.g. "I *feel* love (etc.) for Janet."

3. A TAXONOMY OF EMOTIONS

Previous writers have, of course, made distinctions between the various emotions. Sloman, for instance, distinguishes Anger, Exasperation, Annoyance, and Dismay. Nevertheless, different people often draw slightly different divisions between such related emotions, and doubtless these differences depend on the community one grows up in. For instance, 'being emotional' about something is not (for me) quite the same as 'having an emotion' about it. To my mind, the former is a comment primarily on the subject's behaviour, while the latter implies that the subject has a definite emotion in mind (jealousy, for example).

Again, 'being emotional' can mean a disposition, attitude or mood, or could be a matter of temperament or character. These are different things again from "occurrent emotions"--emotions 'in the act'. Of course, dispositional usages of emotion words depend on the occurrent usages we have been discussing so far. This distinction is well understood.

De Rivera[3] presents a taxonomy for emotions which he claims is fairly comprehensive, and shows the relationships and contrasts between the different emotion words. It is based on a structural analysis built on the concept of psychological distance in interpersonal space. An emotion involves a movement--a push or pull--between the subject and an object or 'other'.

The movement or force may be *towards* or *away*. The movement may be *of* the *Subject*, or of the *Object*. In either case, the one moving may be initiating the motion (active) or it may be forced by the other (passive). The movement may be in any one of the *dimensions* of Belonging, Recognition, or Being. Figure 1 from [3, Fig. 11] summarises the overall scheme.

De Rivera explains his three dimensions in terms of James' [9] notions of the 'Self'. The Belonging emotions relate to one's 'material' self (one's body, family, home, etc.). The Recognition emotions relate to one's 'social' self - image in the eyes of others. The Being emotions relate to one's 'spiritual' self (one's spirit, soul, mind, psyche).

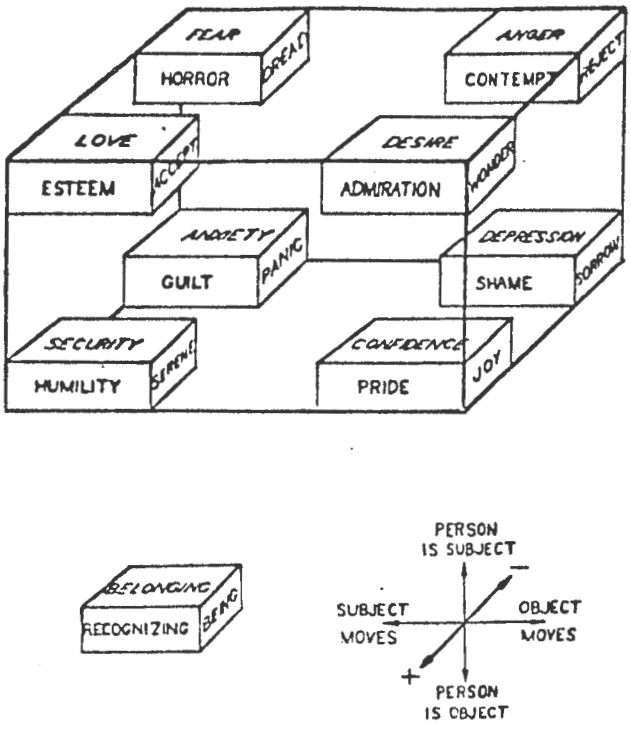


FIGURE 1. Matrix of Emotions (from de Rivera [3])

This structural analysis is very interesting, yet has an air of considerable formal structuring. One fears the word meanings might have been forced into the mould of a neat diagram. However, de Rivera cites some confirmation of this taxonomy [*op.cit.*]. The 'sites' in the cubic diagram represent 'pure types' of emotion, and have been labelled with words generally representative of their kinds. He claims that individual emotion words have meanings which tend to cluster into one or another 'site' in the diagram

His continued analysis surveys how some of 100-plus different emotion words in use fit into this scheme. This shows that in some cases different emotion words occupy the same 'site' yet are distinguished in everyday use. Thus further distinctions must be made in the structural apparatus.

The one extra distinction that appears to be fairly clearly established is in the dimension "fluid"/"fixed". For example, both Anger and Hate occupy the site for movement By the Subject in the Belonging dimension Away from the object. However the behaviour evinced in an angry person does something to resolve the emotional feelings ('catharsis') while 'hate' is in a sense 'frozen' and is not relieved by outward action. (The most one can do is to turn ones attention away from the object of the hate, or seek a different perspective so as to change the evaluation underlying the motion. Perhaps eventually the hate can be converted into anger, and then worked out; the fact we can talk about this conversion lends credence to the analysis.)

4. RELATIONSHIP OF THESE THEORIES TO AI

Following Sloman [1, 10] we can adopt a model of mind architecture involving a database of information (facts, tied into models of the world and of the self—many partial models actually) and databases (stores) of current purposes, goals, expectations, etc. The intelligent being has many purposes at any particular time, only some of which are selected for attention. Purposes may be represented by descriptions of states or events to achieve or to avoid, and are also characterised by descriptions of their relative importance or priorities, and dependencies.

The considerations advanced above lead pursuasively to a *causal-evaluative* theory of emotion. We must now attend to two important components or prerequisites for having emotions: *feelings*, and *purposes*.

In the context of cognitive science, hitherto, *purposes* have received attention, while for the most part *feelings* have not. This is presumably the result of applying the Information-Processing metaphor for mental functioning, which draws attention towards intellectual and rational functions and away from feelings. For instance, feelings are barely mentioned in [10], and are not even listed in the index for [11]. Slater[12] describes entertainingly how our male-dominated culture encourages us to ignore feelings, and other feedback signals, as we pursue our individualistic goals and life scripts.

4.1. Feelings

Somehow, feelings have to be brought into the scope of the information processing model. As a first step, we regard feelings as the psychic or mental counterpart of the fact that intelligent (and stupid) beings have bodies—a physical substrate. In the case of living creatures, at least, the physical body and the mind are related, and evolution (or creation?) has provided feedback loops from body to mind. This feedback is partly through perception mechanisms, such as sight and hearing, and partly through what we call feelings.

Taking a hint from Jungian psychology, we model the influence of feelings in cognitive mechanisms through associating *'feeling vectors'* with *all* mental processes and data in the information processing model. A calculus or theory for how these feeling vectors should be transformed during mental processing remains to be worked out.

The simplest model of this kind would be to associate a single signed numerical "desirability" or "intensity" value with each thought and process. Indeed, this case bears some analogy to the Freudian theory, which sees all mental processes as involving flows (and sometimes captures) of Libido or psychic energy between different parts of the psyche [13]. This is undoubtedly too simple a model, but it meshes quite nicely with other parts of the theory. (It also suggests that a 'conservation law' for feeling vectors or for psychic energy should be sought.)

Sloman and Croucher [1, 10] identify different purposes and different motives as having differing 'priorities'. This 'priority' can be seen as really (part of) the feeling intensity for the purpose or motive. (One can even see how evolutionary mechanisms might favour organisms able to make priority distinctions between conflicting urges.) Similarly, de Rivera has a 'movement' component (towards or away) in each emotion. This again is a (signed) 'intensity', with negative values (by convention, but not arbitrarily so) being for movements *away*.

Again, some other intelligent systems have displayed a need for similar continuous-valued control parameters. Expert systems such as MYCIN use them[14]. Berliner has also demonstrated the need for something like this in game playing programs, with his 'smoothly varying application coefficients' [15].

The Freudian and Jungian theories are united in ascribing all sources of psychic energy to the unconscious, with roots ultimately in the instinctual mechanisms of the organism (those instincts being reflected in 'archetypes' according to Jungian theory). In a computer, we can of course *model* such a theory of feeling.

We claim, nevertheless, that for feeling to be 'real' (whatever that means, depending on your philosophical slant) it must be based at least in part on 'genetically programmed' or 'innate' mechanisms which provide feedback to the mind from its physical implementation body. Thus it will be premature to talk about feelings in robots until they do indeed need to interact with the environment for their 'life support', etc., and have some built-in awareness of this. A robot which periodically has to recharge itself from a wall socket, and the micro-mice in the maze competitions, are a primitive start in this direction. These conclusions are consistent with the arguments of Dennett [16].

Note, however, that we do not consider a single-valued real number associated with each mental datum to be an adequate model of feelings. We suggest that a vector of values, or some other structured data type, would be more appropriate. However, until such models are implemented and tested, there is no evidence to settle the matter.

4.2. Purposes and Motives

Given the understanding that purposes and motives are qualified by feeling vectors, we have little further to add to previous discussions of 'purposes' and 'motives'. Sloman [1, 10] has presented an outline account of how purposes and motives can be implemented under an information processing model of the mind. Boden's [11] position seems to be compatible with this also. She argues that purposes are *rooted* in the mechanistic processes of the brain, but that we are not thereby forced to adopt a reductionist (or determinist or behaviourist) viewpoint. We emphasise more strongly the role of feelings as part of the rooting of mental processes in the physical brain--and in the organism generally.

4.3. Analyses of Emotions and Feelings

We suggest that de Rivera's structural analysis is a fruitful starting point. We shall outline how to reinterpret his model in the concepts of an IP model of cognition (suitably enhanced).

An objection of principle might be made, that the structural analysis involves an assumption that every emotion involves 'an other'. But this is not really a new difficulty; the same assumption is embedded in the view that an emotion involves an evaluation of the situation which includes consideration of the subject's purposes. The three 'dimensions' of Belonging, Recognition, and Being are not readily grasped; they were introduced earlier, and we return to them shortly.

De Rivera makes a distinction between "fluid" and "fixed" emotions. (It is not clear whether this is intended as a discrete binary-valued or a continuous parameter.) We suggest that this distinction can be interpreted in the information processing model as follows. A Fluid emotion is one including consequent motives for action, and if the actions are taken then this removes that psychic energy from the emotion as such (shades of Libido theory!).

A Fixed emotion, by contrast, is one where *no* direct motives are generated (*pace* Sloman). Therefore, naturally, no immediate outlets for the emotion present themselves. (This is not to say a Fixed emotion cannot be resolved in other ways.) The Fixed emotion, in Sloman's terminology, may involve *Motive Generators* and in some later changed circumstances these generators may be able to produce actual motives for action, which can take off some of the bottled energy. The production of actual motives can be regarded as an unfreezing of the emotion, converting it into a fluid one.

Now to reexamine the three dimensions of interpersonal space. We may compare them tentatively with Jung's Theory of Types [17]. Jung classifies people's temperaments (and by implication their emotions and feelings) on four different dimensions or 'functions': the Feeling function (sensitivity to *values*); the Thinking function (orientation towards intellectual and rational thought); the Sensation function (awareness of physical sensations and perceptions); and the Intuition function (sensitivity towards hunches and intuitions which well up from the unconscious mind). Each of these functions is present in a person, though one or another may be stronger.

This theory has been criticised that it may owe more to an attractive 'mandala' representation than it does to observation. Nevertheless, it seems to have some descriptive value with human beings.

The Dimensions of interpersonal space (*a la* James[9]) can be roughly mapped onto Jung's 'functions' as follows. The Being emotions, and spiritual self, are roughly equivalent to the Intuition function. The Belonging emotions and material self are roughly equivalent to the Sensation function. The Recognition emotions and social self are roughly equivalent to the Feeling function. Jung's Thinking function is not involved here, in this view.

The identifications suggested in the previous paragraph are inevitably only tentative, at this stage, in the absence of working computer models of emotion according to these lights. The whole of Section 4 above is intended to show how to modify the "classical" Information Processing model of cognition (as expounded by Sloman and others, in various forms) to take account of the phenomena of feelings and emotions.

5. SUMMARY

A wide range of theories about the nature of Emotion has been surveyed. The causal-evaluative theory appears to be the most adequate approach, and it can be interpreted in terms of an information processing model on the general lines suggested by Sloman and others. The information processing model as described hitherto must, however, be extended by inclusion of a notion of *feeling*.

We discuss feelings, and describe how they can be integrated into an information processing model of mind. They underpin any adequate description of *motives* and of *purposes*. With this addition, we have an approach to a more adequate model of *emotion*. We also draw on de Rivera's structural analysis of emotion, and integrate also elements of the Jungian and Freudian theories of the psyche. Our goal is to develop a unified theory of feeling, emotion, and motivation.

In summary, we consider that Feelings are essential for an adequate theory of mind, that they do not exist in a 'pure intellect' but depend on interaction between the body and mind, and that without them it is inappropriate to talk of Emotions. An intelligence which has a body, and feelings, and purposes, will be able to have emotions and indeed will be almost human (or at any rate animal).

ACKNOWLEDGEMENTS

This research was supported in part by the Natural Sciences and Engineering Research Council of Canada through an operating grant. I acknowledge also conversations with many colleagues, especially those I worked with formerly in the School of Artificial Intelligence at Edinburgh. Naturally, none of them are responsible for any blunders herein. Names are too numerous to list exhaustively, but Professor H C Longuet-Higgins deserves mention for starting me on Kant's *Critique of Pure Reason* and Aaron Sloman for being provocative. Wittgenstein didn't make it into this paper.

REFERENCES

[1] A Sloman & M Croucher, *Why robots will have emotions*. Proc. 7th IJCAI, UBC, Vancouver, pp.197-202, 1981

[2] W Lyons, *Emotion*. Cambridge Univ. Press, Cambridge. 1980

[3] J de Rivera, *A structural theory of the emotions*. Internat. Univs. Press, New York, 1977

[4] H C Wyld (ed.), *Universal dictionary of the English language*. Routledge & Kegan Paul, London, 1961

[5] K T Strongman, *The psychology of emotion*. J. Wiley, NY, 1973

[6] R W Leeper, *A motivational theory of emotion to replace "Emotion as disorganized response"*. Psych.Rev. vol.55, pp.5-21, 1948

[7] R W Leeper, 'The motivational theory of emotion' in *Understanding human emotion*. (C L Stacey & M F DeMartino, Eds.), Howard Allen, pp.657-65, 1963

[8] R L Gregory, *Eye and Brain*. McGraw Hill, New York, 1973

[9] W James (1890), *The Principles of Psychology*. Dover, New York, 1950.

[10] A Sloman, *The computer revolution in philosophy*. Harvester Press, Hassocks, 1978

[11] M A Boden, *Purposive Explanation in Psychology*. Harvard Univ. Press, Cambridge, Mass. 1972

[12] P Slater, *Earth-Walk*. Bantam Books, Toronto, 1975

[13] S Freud, *New Introductory Lectures on Psychoanalysis*. (transl. J Strachey) Penguin Books, Harmondsworth, 1973

[14] E H Shortliffe, *Consultation Systems for Physicians: The Role of Artificial Intelligence Techniques*. Proc. CSCSI Conf. May 1980, Victoria BC, pp.1-11 (1980)

[15] H J Berliner, 'Computer Backgammon', *Scientific American*, vol.242, 6, 64-72, 1980

[16] D C Dennett, 'Intentional Systems', in *Mind Design*. (ed. J Haugeland), MIT Press, Cambridge Mass., 1981

[17] C G Jung, *Psychological Types*. (transl. H G Baynes, revised R F C Hull) Bollingen Series, Princeton Univ. Press, 1974

EVENT-BASED ORGANIZATION OF TEMPORAL DATABASES

Sanjay Mittal

Department of Computer and Information Science
The Ohio State University

ABSTRACT

Many problem-solving applications need information about temporal properties of objects in the domain of application as well as test for patterns of temporal relationships between different objects. In the approach presented here temporal information is organized in the form of events which describe some temporal property of an object. In this scheme, events are organized around some 'key-events' in a hierarchical fashion to form 'episodes'. Continuous events are modelled by an episode of related point-events. Recurring episodes form an episode-cluster. This scheme also handles the imprecision and variety in temporal description in a natural manner. Finally, some strategies are discussed for answering many typical temporal queries in the context of a particular application of these ideas for organizing temporal data in the medical domain.

INTRODUCTION

In many domains, solving problems or providing expert-like consultation not only requires a database of facts about objects in the domain but also information about temporal variations in some attributes of these objects. Examples of such domains include clinical decision-making, fault-diagnosis of electrical or mechanical systems, economic forecasting, and monitoring changes in the state of some system. In medical diagnosis, for instance, one not only needs to model the state of the patient at the time of diagnosis but also changes in the various signs, symptoms, and labdata over a period of time. In addition, many diagnostic rules are based on the temporal and causal relationships between different data concepts.

The temporal information required to model the dynamics of the objects in the domain of such applications is typically event-oriented. In other words, the dynamics can be adequately represented by describing what events occurred; when they occurred; how long they lasted, if indeed it is useful to represent their duration of occurrence; and how they were related temporally and causally. Given such a database, the typical temporal questions are of the following kind: (a) What events occurred at time T? (b) Did event Y occur? When? (c) Did event Y occur before or after event Z? How far apart were they? (d) What was the change in some attribute of an object during interval T? (e) When did event Y start? How long did it last?

In this paper, we describe an event-based organization which can be used to represent the different kinds of temporal information and answer a wide range of questions based on that information. This organization is based on the following observations. First, events which can be conveniently viewed as having occurred at the same time can be grouped together into an event-cluster. Because of this grouping, reference to an event implicitly refers to the event-cluster as well to which an event belongs. For this reason, we shall use the terms event and event-cluster interchangeably, except where the distinction is important. Second, events are typically organized around some 'key' events, in effect forming a group of event-clusters related temporally. The events in each such group are related to each other by certain criteria. These groups will be considered as episodes. Third, description of temporal relationship between events is often imprecise, incomplete, or both. The implication of this observation is that some questions may not be answerable at all and others only partially, in any given state of the database, though availability of more precise information in the future should lead to a re-organization and possibly more accurate answers. Fourth, the time when an event occurred can be described in a variety of ways and it should be possible to integrate them all in a single organization. Fifth, there is an inherent duality between interval-based and point-based representations of temporal information. In other words, it is possible to 'collapse' intervals into point-events and vice-versa (where applicable) depending on the questions asked and specificity desired in the responses.

These observations point to the need for a set of possibly inter-linked episode clusters. The questions described earlier can all be answered by suitably searching in this largely hierarchical organization of events. We have

used this approach to represent a variety of medical data about patients which can then be used by an automated medical diagnosis system. The two systems, PATREC which organizes medical data about patients and MDX which provides clinical consultation using this data, are described elsewhere [Mittal and Chandrasekaran 1980, Mittal et al 1979, Mittal 1980]. Here we will focus only on those aspects of the database organization which pertain to temporal information.

REPRESENTATION OF TEMPORAL DATA

We shall discuss our approach to representation of temporal data in the context of examining the nature of description and use of temporal information in applications such as clinical decision-making or trouble-shooting complex mechanical systems. For illustrative purposes we will use the medical domain though similar issues are relevant in other domains as well.

Description of Events

An event can be viewed as an assertion about the occurrence of some data concept. For example, in two weeks after admission, the patient had intermittent jaundice, one of the events is 'patient had intermittent jaundice' which is an assertion about the data concept 'jaundice'. Events occur at some point in time which we shall call the temporal descriptor of an event. Thus in the above example, 'two weeks after admission' is the temporal descriptor for 'intermittent jaundice'. Note that the temporal descriptor not only describes when an event occurred but also serves to uniquely identify it (of course only to the extent that the temporal description itself is unique). For example, in 'jaundice at admission' and 'jaundice three days after surgery', the temporal descriptors "at admission" and "three days after surgery" not only describe the time of occurrences of the two jaundice events but also distinguish between the two events. There are some interesting issues about what happens if these descriptors refer to the same event (this may be determined from other information) or if a temporal description does not identify a unique event-cluster. We shall postpone a discussion of these issues to a later report [Mittal 1982].

Also note that often the time of occurrence of one event is described in terms of another event. In the previous example, the temporal descriptor for the event 'intermittent jaundice', namely, "two weeks after admission" was in terms of another event 'admission'. In general, the notion of events and their temporal descriptions will be relative to one another. A particularly useful instance of this relativity occurs in the case of certain events whose time of occurrence is given in terms of the event itself. As we point out later, this self-referentiality is often an important clue that these events play a important role in the domain for organizing other events.

Organization of Events

Event Clusters. Events which are viewed as having occurred at the same time, i.e., defined by the same temporal descriptor, can be grouped into an event-cluster. This grouping is actually an artifact, albeit a very useful one. It allows extraneous temporal information to be ignored because even though events within an event-cluster may have occurred hours or even days apart, they can be usefully viewed as occurring at the same time. This 'collapsing' of an interval into a point-in-time is essential in managing the large amounts of temporal information. The clustering may be performed statically by the user, i.e., a set of events may be given to the system as an event-cluster (with a single temporal descriptor), although, in reality, the events may have occurred at different times. This clustering may also be performed dynamically by the system in response to certain questions. One of the interesting characteristics of our scheme is this ability to maintain the duality between point and interval-based representations, i.e., allow temporal descriptors of events to be described to any degree of specificity but also allow an interval to be viewed as a single 'point in time' for answering some questions. We will discuss this issue in greater detail later on.

There is an important consequence of this clustering of events. A temporal descriptor of any event in a cluster also describes the time of occurrence of any other event in the cluster. In other words, uniquely specifying an event in a cluster also specifies the cluster. For example, given the following cluster of events,

> "at admission the patient had jaundice, pruritus, abdominal pain, and complained of nausea and vomiting",

the temporal descriptor 'at admission' not only describes when the patient had jaundice but also pruritus, nausea etc. Furthermore, a new temporal descriptor 'at(jaundice at admission)' can be created from one of the events in the cluster, namely, jaundice, which serves the same function as the previous temporal descriptor. Thus, we shall use the term event to refer to event-clusters also in the rest of the paper, keeping in mind that an event implicitly also refers to the event-cluster to which it belongs.

Grouping related events into episodes. Given these event-clusters, how do we represent the temporal relationships between them? Typically, these events do not exist in a linear temporal relation. There are various reasons for this. First, the temporal descriptions defining an event are imprecise or incomplete. For example, "two weeks after admission" does

not mean precisely 'two weeks' but only approximately so. Similarly, in the following data about a patient,

> "...a week after surgery, the patient complained of severe pain in the abdomen, nausea, and vomitting. Three days after admission, he had high fever and complained of chills...",

'a week after surgery' is imprecise. But the relation between this description and 'three days after admission' is not even known, unless one makes the inference that the 'admission' being described must have taken place after the event 'severe pain', in which case the relationship is known but still imprecise. In this example, there are two event-clusters, "a week after surgery" (abdominal pain, nausea, vomitting, etc.) and "three days after admission" (fever, chills, etc.), with the latter occurring after the former.

Second, new events are described in terms of existing events. For example, in the above fragment, 'a week after surgery' was derived from surgery, 'three days after admission' was derived from admission, and admission was, by inference, 'after a week after surgery' or more precisely 'after the event severe pain which occurred a week after surgery'. Thus, there is a natural grouping of events around ones which are used to describe them.

Finally, there are some events which are viewed as 'key' events, in the sense that many events are grouped around them. These are events which are important to remember (and other events are remembered based on them) based on domain-dependent criteria. For example, in the medical domain, 'admission', 'discharge', 'surgery', etc. are some of the key-events. An important property of these events is that they can be self-referential in the sense defined earlier. One consequence of this property of key-events is that large sub-sets of temporal data can be usefully organized and retrieved around these key-events without knowing precisely (or at all) how these events are related to other events in the database. For example, large amounts of data about a patient can be organized into event-clusters around the key-event 'admission', and useful diagnosis performed, without ever knowing when this admission event actually took place.

We shall refer to these groups of events organized around some key-event as episodes. In general, an episode can be viewed as a partitioned hierarchy of event-clusters, originating from the key-event cluster. Each level in the hierarchy contains events directly defined with respect to the event at the next higher level (let us call this event the d-event, short for the 'defining event'). The left partition contains events which occurred before the d-event and the right partition contains events which occurred after the d-event. Note that the partitioning is only with respect to the d-event at that level and not the key-event at the top of the hierarchy. One implication of this relative grouping of events (i.e., relative to the d-event at each level) is that some events in the left partition might actually have occurred after the key-event defining the episodic hierarchy. The converse may hold true for the right-partition. Thus for efficiently answering questions such as "Did X occur before/after event Y?", the episodes might also be organized into a separate hierarchy: here events at one level are strictly before or after the events at other levels to which they are linked and different events at the same level are temporally indeterminate with respect to each other. (By temporal indeterminacy we mean the following: due to a lack of precision in temporal descriptions, it may not be possible to say which of the two events in question occurred before or after the other one.) This latter hierarchy is a hierarchical representation of what may be thought of as the time-graph of events in an episode. The former hierarchy can be thought of as the definition hierarchy of events in an episode. The latter can be created from the information contained in the former.

Let us illustrate this organization with an example. Consider this somewhat complex description:

> "A month before admission, the patient first complained of fever and chills. He took medication and felt better. A week later, the fever recurred, accompanied by anorexia. A few days later, he developed jaundice. At admission, he was found to have jaundice, pruritus, anorexia, severe abdominal pain, etc. Two months before admission, he had eaten shellfish and had vomited a few times the next day. A few weeks later, he had an attack of acute abdominal pain, but it subsided. Three days after admission, his labtests were ... . Two days later, he was operated upon for gallstones. A few days after admission, he was alright and discharged."

The key-event for organizing all this data is 'admission'. All other events can be directly or indirectly grouped around 'at admission'. The two hierarchies - definition and time-graph - for this episode are shown in figures 1 and 2 respectively. A few explanatory comments are in order here. Our current implementation does not parse the natural language text as presented. Instead, the information contained in the text has to be given to the system in a stylized format, one event-cluster at a time with each cluster preceded by the temporal description of the cluster. Figure 3 shows the actual format for entering some of this data. However, not processing the actual text prevents the system from making certain inferences which would help in reducing some of the imprecision in the temporal description of events. For example, by

itself it is hard to say if the event E9 (refer to figure 1) occurred after the event E8 or even E7. This is because of the inherent imprecision in the phrase 'a few days after admission'. However, the textual occurrence of E9 after E8 and E7 coupled with the fact that 'a few days' can be more than 3 or 5 days, can be used to infer that E9 actually occurred after E7 and E8. Other inferences are also possible based on domain knowledge. For example, a property of the event 'admission' in the medical domain, namely, patients are admitted for problems originating before admission, can be used to decide that event E10 must have occurred before E1 (i.e., admission) even though the phrase 'a few weeks later'

Other criteria for creating episodes. So far we have described only one criteria for grouping events into episodes, namely, grouping around some key-event. There is often need for other types of episodes as well. For example, events with duration can be modelled as an episode about the occurrence of the event. These episodes would contain information about the event describing the start of the event, events describing changes in the state of the concept underlying the event, the termination of the event, other events which may be closely linked, etc. Consider the following description:

> "Jaundice was first noticed a week after surgery. It gradually increased in intensity, till the patient was given medication four days later. It subsided, but recurred a month later when the patient was admitted. Three days later, he was operated a second time. There was no jaundice a day after surgery"

This description can be modelled as an episode for jaundice (in addition to the other episodes based on key-events), but one in which only events pertinent to jaundice are stored. Such an episode allows questions about jaundice to be efficiently answered, as well as allows jaundice to be treated as a point-event in the other episodes. We shall discuss this in more detail in the next section.

We are currently investigating what other criteria are useful for episodic groupings. This and related issues are dealt in a forthcoming report [Mittal 1982]. It should be emphasized that episodes other than those based on key-events are meant to provide additional organization for efficiently answering certain kinds of questions, in particular, questions about continuous events and causal relationships. The key-event based episodic groups contain all the necessary information for answering such questions, though not quite as efficiently if these additional structures are not imposed.

Relationship between episodes. An episode may be linked to another episode either directly by means of a known temporal relationship between the respective key-events or indirectly because of a relationship between events in the two episodes. For example, in the description for jaundice given above, there are three episodes: "at first surgery", "at admission", and "at second surgery". The relation between the last two episodes is direct, namely, "at second surgery" was "three days after(at admission)". But the relationship between the first two episodes is more indirect, namely, "at admission" was "1 month after" the event "4 days after(1 week after(at first surgery))".

There are other reasons also for relating different episodes. One particularly important one is between different episodes of a recurring event. For example, different admissions, different bouts of jaundice, etc. We shall refer to groups of episodes based on recurring episodes as episode-clusters. Lack of space does not permit a comprehensive discussion and interested readers should see [Mittal 1982].

Variety of Temporal Description

As the examples discussed so far show, temporal descriptions can take many forms. Some of the different kinds found in the medical domain are: (1) key events, such as 'admission'; (2) those derived from key events, such as 'three months after second admission'; (3) those derived from other events, such as 'a week before jaundice', or 'a month after the last time he was administered halothane'; (4) those contexually derived in a narrative, such as 'four days later...', 'a month earlier', or 'two weeks after the previous surgery'; (5) and calendar events, such as 'March 1975', 'January 15, 1956, 4:56pm' or 'latter part of 1981'. All of these different kinds are frequently used to describe the time of occurrence of events, all can be more or less imprecise and ambiguous, and they require different kinds of knowledge to understand and integrate.

Aside from requiring a model which can integrate all these different kinds of descriptions, this variety also creates potentially conflicting multiple temporal descriptions of the same event. Issues pertaining to inconsistency or ambiguity in temporal relationships are beyond the scope of this paper. Interested readers are referred to [Mittal 1982].

EPISODIC ORGANIZATION IN THE PATREC SYSTEM

In the earlier section, we have described an episodic organization for the representation of temporal data. To recapitulate, temporal data are represented as events. Events occurring at the same time are clustered together and described by the same temporal

descriptor. Event-clusters are further grouped into episodes, often organized around some key-event. Finally, episodes of recurring events are clustered into episode-clusters.

We have implemented some of these ideas in the temporal component of the medical database system (PATREC, see [Mittal and Chandrasekaran 1980]) and the following discussion will be in the context of the PATREC system. The important things to know about the PATREC system are: it has a conceptual model of medical data, organized as a hierarchy of frames; and the actual data about a particular patient are stored as a patient model by instantiating the relevant frames from the conceptual model.

Episodic Organization of Patient Model

Each instance of a medical data concept can be viewed as an event. Events that are described as belonging to the same event-cluster are grouped in an event-cluster frame. Each event-cluster frame has an associated temporal description frame which defines the event-cluster. Each event-cluster is organized in one of the hierarchies - one for each key event described for a patient. An event may be implicitly linked to more than one hierarchy because of the multiplicity of descriptions. Sometimes events may be described which cannot be linked to any of the key-episodes (i.e., episodes formed from some key-event). In such cases, the defining event is used to create a temporary key-event around which an episode may be formed.

Representation of Imprecision

In our current implementation we have adopted a relatively simple representation of the imprecision of temporal description, which nevertheless has proved to be quite powerful. Our model allows for both implicit and explicit imprecision. The former refers to the observation made earlier that temporal descriptions are inherently imprecise. For example, given 'one year ...', it is represented as ((YEARS 1) (MONTHS U)(DAYS U)). Another heuristic that is useful for implicit fuzziness, assumes that the higher units of time, if not specified, are zero. For example, 'three months...' is represented as ((YEARS 0)(MONTHS 3)(DAYS U)).

Sometimes the range of imprecision is made more explicit. For example, 'between two and three years...' will be represented as ((YEARS (BETWEEN 2 3))(MONTHS U)(DAYS U)).

During the process of searching the temporal structure for answering questions, these imprecise descriptions are combined using some simple heuristics. One such heuristic states that, 'if the highest precision in one descriptor falls within the imprecision of another, then the new description is no more precise than the second one'. For example, combining 'one year' and 'two months' still yields 'one year'. Needless to say, such heuristic combinations can easily introduce errors. However, in the medical domain our experience indicates that imprecise events are combined (or compared) not for obtaining an exact answer but mainly for relative order, a purpose adequately served by our heuristic approach.

In order to complete the discussion of this model, let us consider how some typical questions can be answered using the organization outlined above.

When type questions

One common type of questions are of the form, "when did event Y occur?". Examples of such questions are, "when did the patient have jaundice?", "when was the first occurrence of pruritus?", "when was liver surgery performed?".

The basic strategy for answering such questions involves a search of the different episodes in which the event could have occurred. For all such episodes selected, the search begins at the key-event defining the episode. Thus, questions about key-events can be answered most efficiently. (This could be viewed as another definition of key events.) As mentioned earlier, episodes may be created based on other criteria as well. Therefore, in case of event types for which such episodes exist, such questions can again be answered efficiently. In general, however, the temporal organization is not sufficient by itself for efficiently answering when-type questions. A variety of secondary access structures need to be created. These structures capture patterns of temporal relationships inherent in the domain. For example, direct links may also be kept for certain events which are frequently queried about. These allow questions such as "Did X occur?", "When was the first(or last) occurrence of X?" etc. to be answered without a great deal of search. Certain data concepts have constraint expressions which limit the episodes in which events about these concepts can occur. For instance, labtests are only performed after admission. Thus given a question such as 'When was SGOT performed?', the system need only search the time-graph hierarchies in the admission episode-cluster and then only the right partitions. Most of these secondary access links are domain-dependent and beyond the scope of this paper.

Questions over a time interval

Even though the organization is largely event-based, it allows for the creation of time intervals and asking questions over that interval. Examples of such questions are "did the patient have pruritus in the year preceding admission?", "did he have abdominal pain between

the onsets of jaundice and pruritus?", "how long did the jaundice which started a week prior to admission last?". The first two require the system to create an interval and search it in the required order. Answering such questions is more efficient than the general when-type questions because here the specification of some interval provides a more bounded temporal organization. In medical diagnosis, at least, such bounded when-type questions are much more frequently asked than the general ones. In particular, once a certain interval is specified, a large numbers of questions are asked in that interval. For example, the typical diagnostic scenario involves specifying a 'window' around some key-event and then querying the database about the occurrence of large number of medical data objects.

The strategy for answering the third type of question mentioned above, namely, "how long did jaundice which started a week prior to admission last?" depends on how a particular continuous event is modelled in the database. Some events with duration are modelled as separate episodes. In such cases, most of the work is in determining the relevant episode in the episode-cluster for that particular data concept. For example, in the case of the above question, the first step is finding the episode for jaundice which occurred "a week prior to admission". This can be done by the same strategy employed for bounded when-type questions. Once the jaundice episode is retrieved, it can be easily searched for the event representing the termination of jaundice, and the answer returned. This latter search can often be avoided for some questions by summarizing the important temporal properties from this data-episode into the point-event represented in the key-episode. This also allows a natural way to handle continuous events which have not terminated yet, or summarize a data-episode into a point-event, or summarize an episode-cluster into a single episode. For more details refer to [Mittal 1982].

Representing continuous events by separate episodes is neither required for all data objects nor feasible because of space limitations. Therefore, temporal information about the continuous behavior of those data objects which are queried about less often is represented by point-events describing when it started, when it ended, causal links to other events, etc. In such cases, answering these questions could involve an exhaustive search of the key-episode(s) in which events describing the data object lie unless some reasonable assumptions were made about typical duration of particular events or typical temporal interval between two causally-related events. In our current implementation we expect the data concept behind each event to contain knowledge about typical duration etc. to limit the search.

Comparing two events

One of the most important type of questions that are asked are of the form, "did event Y occur within interval T of event Z?". Such questions are very useful in establishing causal links between events. Here we describe some of the basic strategies for such questions. For more details see [Mittal 1982].

(1) If the two events have associated calendar times, then use them, unless the precision is less than specified in interval T, or the precision does not allow a comparison.

(2) In some cases, events Y or Z may be defined in terms of the other. This relationship would be used with the caveats noted above.

(3) If both events are organized in the same episode, then try to find a common event to which both can be related. Again the same caveats apply.

(4) If the events are organized in different episodes, first try to relate each event to the key-event of the episode. Next, try to relate the two key-events. Again the same caveats apply.

It should be pointed out that the lack of precision and completeness in temporal description implies that not all such questions can be answered and some may be answered at best as 'maybe'.

REVIEW OF RELATED WORK

The little work that has been done in representing and reasoning with temporal knowledge can be categorized on a spectrum ranging from domain-dependent to domain-independent. By domain-dependent we basically mean that the temporal structure and reasoning are strongly guided by the content of the objects in a particular domain. An important work of this type is Kolodner and Schank's model for organizing events along the lines of human memory organization. The CYRUS database system [Kolodner 1978], which is based on these ideas, is organized using a semantically-rich model of objects in the world of diplomacy such as political personalities, roles, places, events, etc. Questions about events and relationship between events can be answered by exploiting a rich interconnection between different types of events. Our episodic model was also influenced by Kolodner and Schank's work.

One of the earliest attempts at building a domain-independent time-expert was the work by Kahn and Gorry [Kahn and Gorry 1977]. Some of their analyses regarding description of events, and representation of fuzziness have proved

useful in our work. More recently, Allen [Allen 1981] has proposed an interval-based representation of temporal knowledge. The notion of 'persistence' of events and use of intervals to represent imprecision in event description may prove to be useful in organizing temporal databases too.

The approach presented in this paper differs from the earlier ones in developing a framework in which domain knowledge can be easily integrated into a largely domain-indapendent scheme. As we showed, our approach allows a variety of temporal knowledge to be efficiently organized and queried about regardless of the domain being modelled. Another interesting property of our scheme is the ability to maintain the duality between point-based and interval-based representation of events and make a transition from one to the other depending on the questions asked. Intervals are represented in the progression from events, to episodes, to episode-clusters. Intervals can be collapsed by the use of event-clusters, by summarizing data-episodes into an event, and by summarizing episode-clusters into episodes. Finally, our scheme allows for domain knowledge in the form of key-events, constraints on temporal relationships between events, typical behavior of continuous events, etc. to be integrated with the other domain-independent mechanisms for better performance.

Another interesting work, though orthogonal to the work reported here, is the RX system [Blum 1981]. In this system, a temporally organized database is used to infer new relationships between medical data, that is, temporal relationships are used to infer causal relationships.

SUMMARY

We described an episodic event-based organization of temporal databases that are typically used in many expert-system applications. The organization allows a large number of events to be represented and efficiently queried about. It was shown how a variety of temporal descriptions can be integrated into a single organization, even when such descriptions are imprecise and incomplete. A wide range of questions were considered which can be easily answered using this scheme. An interesting feature of the organization is the ability to answer interval-type questions in a natural manner using essentially an event-based representation. This points to the duality of the interval-based and point-based representations. Finally, some details of the actual implementation were discussed which incorporates both domain-dependent and domain-independent aspects of temporal knowledge.

ACKNOWLEDGEMENT

I am very grateful to B. Chandrasekaran for his many suggestions and in helping to shape the ideas presented here. The work described here has benefited from many discussions with other members of our group, in particular, Ashok Goel, Jon Sticklen, and Jack Smith. It was supported in part by the grant MCS-8103480 from the National Science Foundation.

REFERENCES

Allen, J. F., "An Interval-Based Representation of Temporal Knowledge", Proc. Seventh IJCAI, Vancouver, Canada, August, 1981

Blum, R. L., "Discovery and Representation of Causal Relationships from a Large Time-Oriented Clinical Database: The RX Project", Ph.D. Thesis, Interdisciplinary Program in Computer Science and Biostatistics, Stanford University, 1981

Fries, J. F., "Time-Oriented Patient Records and a Computer Bank", In J. Am. Med. Assoc. Vol. 222, 1972, pp. 1536-1542

Kahn, K. M., and G. A. Gorry, "Mechanizing Temporal Knowledge", Artificial Intelligence, Vol. 9, 1977

Kolodner, J. L., "Memory Organization for Natural Language Data Base Inquiry", Research Report 142, Dept. of Computer Science, Yale University, September 1978

Mittal, S., B. Chandrasekaran, and J. Smith, "Overview of MDX - A System for Medical Diagnosis", In Proc. Third IEEE Annual Symposium on Computer Applications in Medical Care Washington D. C., October 14-17, 1979

Mittal, S., and B. Chandrasekaran, "Conceptual Representation of Patient Data Bases", in Journal of Medical Systems, June, 1980

Mittal, S., "Design of a Distributed Medical Diagnosis and Database System", Ph.D. Thesis, Dept. of Computer and Information Science, The Ohio State University, 1980

Mittal, S., "Organization, Inference, and Retrieval of Temporal Information in Support of a Decision-Making System", in preparation.

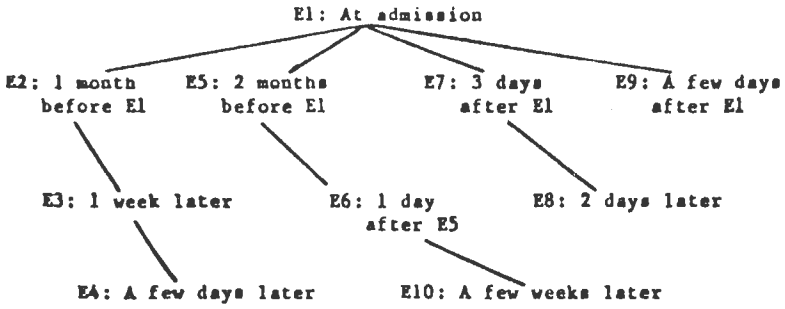


Figure 1. Definition hierarchy of the episode

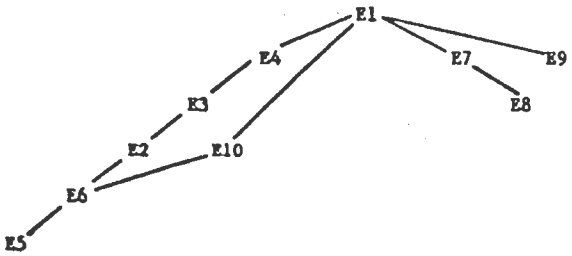


Figure 2a. Time-graph without inferences

```
TIME>>(AT ADMISSION)
        D>>JAUNDICE
        D>>PRURITUS
        D>>(PAIN ABDOMEN SEVERE)
        ...
TIME>>((2 MONTHS) BEFORE ADMISSION)
        ...
TIME>>((3 DAYS) AFTER ADMISSION)
        ...
TIME>>((2 DAYS) LATER)
        ...
TIME>>((U DAYS) AFTER ADMISSION)
        ...
```

Figure 3. Sample of actual data entry to PATREC

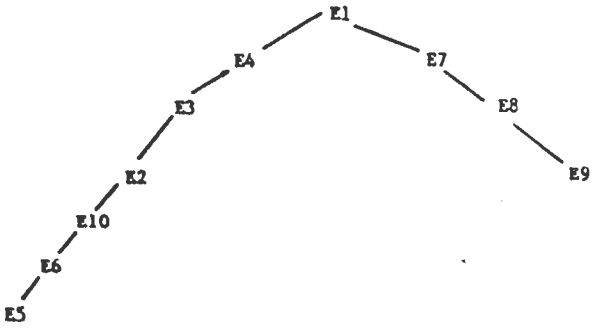


Figure 2b. Time-graph with inferences resolving imprecision

KNOWLEDGE REPRESENTATION AND DATA BASES:
Science or Engineering

Nick Cercone

Computer Science Department
Simon Fraser University
Burnaby, British Columbia, Canada

Randy Goebel

Computer Science Department
University of Waterloo
Waterloo, Ontario, Canada

1. Introduction

A suitable representation can render difficult problems manageable. Artificial Intelligence [AI] researchers are preoccupied with knowledge representation for this reason. In addition, data base management [DBM] researchers realise that large quantities of physical data can be administered more efficiently if there is a logical content or relationship amongst the data atoms (Chamberlin, 1976). The rules for interpreting a data base via a data model, i.e., data semantics, has become of increasing interest to data base managers.

The extent of the divergence between artificial intelligence knowledge representation [AIKR] and conventional DBM over the logical content of data is indicated by their fidelity to the classical idea of deduction. AIKR is relatively faithful to logical deduction, with the exception that denotational semantics has not been universally adopted (McDermott, 1978b). From its inception AI (and AIKR) has been concerned with 'data models' which permit logical deduction usually as an analogue for human reasoning. For AIKR, the need to represent and manipulate more complex and abstract data structures has motivated the development of more elaborate data semantics and accompanying deductive mechanisms.

Traditionally, DBM has been concerned with data independence to the exclusion of logical deduction. 'Data definitions' or 'schemas' have been developed which can describe any collection of source data. As data base administrators become more ambitious about capturing the meaning of their data, they consider epistemologically adequate data models. Logically consistent data semantics must be developed along with epistemologically adequate data models.

Both AIKR and DBM are evolving from heuristic to more formal methodologies. The convergence of the alternative methodologies should occur, we believe, with the gradual adoption by AIKR of the DBM paradigm and the inclusion of AIKR reasoning systems into the DBM paradigm.

2. Artificial Intelligence as an Empirical Study

2.1 The Ad-Hoc Development of Representation Tools

Early AI representation techniques were developed in an ad-hoc manner, largely in response to the constraints of implementation. Question-answering systems developed during the 1950's all lacked a coherent notion of representation. They used a dictionary and a limited predetermined domain of discourse. The question-answering data bases consisted of attribute-value pairs; queries were evaluated by simple pattern-matching according to various criteria, until answers were found.

The conversation machine developed by Green, Berkeley, and Gotlieb's (1959) used attribute-value pairs. Their 'conversation machine' operated in the domain of weather. The data base stored knowledge of the weather which characterized each season and the dictionary stored meanings for ordinary and operator words. Ordinary words included snow, rain, hail, time, today, yesterday, etc. The dictionary represented meanings as attribute-value pairs, e.g., the attribute of a time-classified word is a type of time, calendar or relative, and the associated value is a code for the amount. Operator word attributes consisted of functions and values, supplied when the associated functions were executed. Ordinary words were coded similarly to operator words.

The meaning of a declaration, query, or assertion was represented by the set of codes for the constituent words. Words not in the dictionary were considered meaningless and assigned a code of zero. Otherwise each word was found via table lookup and coded by its attribute-value pair. The conversation machine then compared this 'meaning' with its own store of coded declarations and selected an appropriate response 'frame' from memory (with slots to be filled in with words from the original declaration of query).

The conversation machine limited syntactic analysis to only a few constructions and the problem of selecting the correct response. Encoding terms with attribute-value pairs is still utilised to some extent today.

The numerous question-answering systems built after the 'conversation machine' include the BASEBALL program of Green, Wolf, Chomsky, and Laugherty (1963), SAD SAM from Lindsay (1963), PROTOSYNTHEX of Simmons, Klein, and McConlogue (1964), the 'deductive question-answering system' developed by Black (1968), SIR from Raphael (1968), and the 'semantic memory' model of Quilliam (1968). These systems were limited in representational epistemology. They chose small, well-defined problem domains, relied almost exclusively on the

attribute-value pairs provided by LISP property lists, and utilised clever programming to good advantage. The lack of a coherent representation framework limited further developments in AIKR. Recently AIKR has achieved better understanding of abstract representational primitives. Strategies have been mapped out for property inheritance among concepts, and associative processing algorithms for relating concepts, etc. Now AIKR can apply the representational strategy to a large real world data (knowledge) base.

2.2 The DBM Paradigm and Artificial Intelligence

The data base management paradigm which we believe could significantly contribute to the artificial intelligence treatment of large knowledge bases is presented in Figure 1.

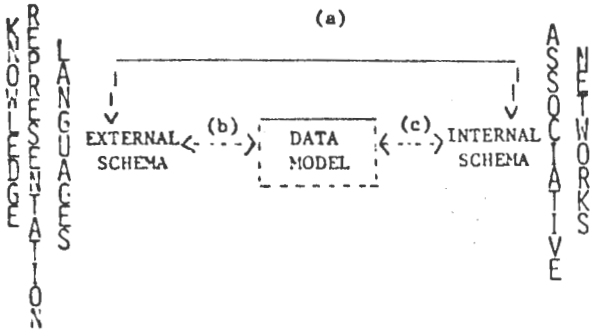


Figure 1.

The data model makes explicit the data interrelationships within the data base. The external and internal schemas represent interfaces between the data model, the user of the data model, and physical devices respectively. Figure 1 illustrates a weak kinship between knowledge representation languages (KRL) utilising frame systems (and scripts, schemata) of AIKR and external schemas and between associative networks (and logical representations) of AIKR and internal schemas. The parallel between the AIKR representations and parts of the DBM paradigm is not exact, but assists discussion of the advantages of using the DBM paradigm for AIKR.

Brachman (1979) classified levels of 'semantic' network representations from implementation to the linguistic level. Adapting Brachman's classification to representation in general results in our summary of representation levels presented in Table 1.

REPRESENTATION LEVELS

Level	Representation Primitives	Examples (non-exclusive)
Implementational	Atoms, Pointers, Arrays, Sets, etc.	Data Structures
Logical	Propositions, Predicates, Arguments, Operators, Functions	Schubert, 1975 Cercone, 1975 Shapiro, 1971 Hendrix, 1975
Epistemological	Concept structures and types, Inheritance and structuring relations, Topic predicates KRL's	Brachman, 1978 Goebel, 1977 Schubert et al, 1979 Bobrow & Winograd, 1977
Conceptual	Semantic, conceptual relations (cases), Primitive actions, States and events, Frames, Scripts	Schank, 1972 Wilks, 1973 Norman et al, 1975 Minsky, 1975
Linguistic	Words, Concepts	Szolovits, 1977 Quillian, 1968

Table 1. Representational Levels.

Obviously these classifications are fraught with fuzzy boundaries; only with hindsight could they have been drawn at all. The ad-hoc development of AIKR representation techniques has led most AI researchers to elaborate their representation level into a 'complete' representation for their theory. Thus it is that semantic networks and frame systems can be thought of as corresponding to either external or internal schemas in the more structured DBM terminology. Although the analogy is imperfect, for purposes of comparison we prefer to think of associative networks (in Schubert's sense) as more closely aligned to internal schemas, and the frame-based representation languages aligned to external schemas. The major missing component for AIKR is the DBM data model. AIKR, in fact, lacks a coherent data model and obfuscates the data model concept by amalgamating the notion under the guise of representation systems, with a corresponding loss of data independence enjoyed by DBM systems. We propose that AIKR capture the data model concept by designing a logical data model, utilising the associative network as an internal schema and a knowledge representation language as the external schema to map user-interface constructions onto the logical data model. Thus far, this functional delineation has not been achieved in AIKR systems.

3. Fundamental Notions in Knowledge Representation

3.1 Knowledge versus Data

Knowledge can be thought of simply as a range of one's information or understanding. Artificial intelligence uses an 'empirical' notion of knowledge: the information which may be accessed by a program is its knowledge. A program's knowledge usually takes the form of an internal representation, organised in ways which are

appropriate for the program's various uses. Some of the major approaches knowledge representation theorists employ to design such programs include semantic networks (Quillian, 1968; Schank, 1972; Anderson and Bower, 1973), logical statements (Sandewall, 1971; Moore and Newell, 1973), and procedures (Winograd, 1972; Hewitt, 1972).

Artificial intelligence programs have, however, traditionally structured information in an ad-hoc manner, in the sense of Lindsay (1973). This structuring is partly the responsibility of the interpreter (program) and partly the responsibility of the experimenter (programmer).

Data, on the other hand, can be defined as factual information used as the basis for reasoning, discussion, or calculation. The concept of a data model in DBM systems separates structure from data; the data model concept is intuitively implied in some AI approaches, for example, productions systems and some frame systems, but has lacked the clarity of the concept as used for DBM.

3.2 Current Issues in AIKR

The designer of a knowledge representation system must face fundamental representation issues very early in his approach to design, and his choices are critical. The designer must decide what should be represented, that is, the content of the representation, the form of the representation, and the level to which the representation is restricted. In addition, since the representation strongly influences the organisation of knowledge, the appropriate organisational strategies and structures must be examined. On the basis of these considerations, we have argued previously in favour of a non-primitive associative network representation in which propositions are organised in normal form determined by the concept hierarchy (Schubert et al., 1979). We believe this representation serves the role of an expressively adequate, semantically well-defined (Tarskian sense) logical level representation. We view this level of representation as analogous to DBM's concept of internal schema and briefly defend that view here.

Woods (1975) believes that 'intensional' and 'extensional' entities must be represented by different sorts of nodes in an associative network. [Extension and intension are terms used to distinguish that to which a designator refers or applies and its meaning]. For example, Woods says that in some contexts "the prettiest blonde" refers to only "Sally Sunshine", yet in other contexts "the prettiest blonde" depends on the notion conveyed by the descriptive phrase. Woods believes that these contexts are distinguished by different sorts of nodes (or sub-networks). We believe that terms (or nodes) already encompass both extensions and intensions, and that a syntactic distinction is not appropriate to distinguish extensional and intensional information. It is appropriate to explain the conditions under which a term contributes to the truth value of a sentence through its intension rather than through its extension alone. We propose that a distinction be made between propositional content and pragmatic aspects. We agree with Woods that internal meaning representations should reflect both propositional content and pragmatic aspects, but the two sorts of information should not be inextricably mixed, to handicap the comprehension processes which must utilise the acquired knowledge. Woods' special syntactic representational device would also encumber the matching process since the matching processes seeking suitable referents of discourse descriptions would depend on the original text. In contrast, Schank (1975) has presented convincing reasons why an internal representation should be in canonical form, relatively independent of the original English sentence. Mingling propositional and pragmatic information about utterances would disperse pragmatic information about a particular section of discourse over the propositional data base. Information about speaker intentions and assumptions would be buried with knowledge about dogs, people, etc. We maintain that a separate model for discourse status (speaker intentions and the like) is necessary. This model is the proper place for semantic information.

We limit our discussion of the form of representation to the issue of property inheritance. Our entended semantic network notation is capable of expressing any arbitrary proposition expressible in English. But any system designed for reasoning about the real world must also effectively exploit property inheritance within generalisation hierarchies. Conceptual entities typically consist of many components, the relationship between these components is valuable information. We require a mechanism which allows inheritance of the relationship from components to corresponding components within a conceptual entity. For example, the attachment relationships between the body parts of birds would require non-trivial inference processes to transfer to other similarly structured animals.

The method of 'variable-sharing' solves this problem and allows for trivial transfer of relationships. Knowledge associated with a generalisation hierarchy is stored as a set of implicative propositions which share one universally quantified node and any number of existentially quantified nodes dependent on the universally quantified node. The antecedents of the implications involve the universally quantified node as argument and correspond to concepts which comprise the generalisation hierarchy. Thus the implicants of a concept are accessible by topic rather than by a long list of propositions involved in the concept, see Goebel (1977). This mechanism facilitates addition of new information and we speculate that it is possible to organise other than monadic concepts, say relational concepts, hierarchically as well.

To argue against the use of a small number of very general primitive predicates for representing meaning in natural language utterances, we contrast the methods of Schank (1972) and Wilks (1973) with the network-oriented state-based representation, Schubert et al. (1979). Wilks is,

perhaps, the most vocal advocate favouring the use of a small number of primitive predicates to represent meaning. Indeed his representation is based almost entirely on approximately sixty primitive predicates. In a recent position paper, Wilks (1977), appears to soften that stance, yet remains vague. In particular, Wilks cites Hayes (1974, 1977) for presenting a number of sophisticated (radical and non-radical) arguments against the use of semantic primitives as Schank and Wilks use them. Wilks writes:

> "One aspect of these criticisms is not radical—in the sense of questioning the very basis of primitives —but it is a demand by Hayes that primitive systems give a more explicit account of the rules regulating inferences concerning a primitive for substance, like STUFF. This demand for greater explicitness is a good one, though there is reason to doubt that any coherent and consistent metaphysics of substance can in fact be given. Two and a half millenia of philosophy have failed to provide one, yet throughout that time everyday conversations about substances, such as coal, oil, and air goes on unimpeded. It is important to stress this fact, so as not to fall into the error of imagining that language about substances requires such a metaphysics of substances in order to function at all. It clearly does not."

Wilks misinterprets Hayes' remarks when he ascribes to Hayes the belief that a coherent and consistent metaphysics for STUFF is necessary for all ordinary language comprehension. At the other extreme, embedding the minimal content of terms into a minimal conceptualisation does not facilitate the human interpretive process. The original term itself suggests what content we could infer in addition to the minimal content. This idea of inference can be efficiently programmed in a semantic structure by inserting probable inferences with direct reference to the word definitions. This is simpler than analysing the minimal representation and then looking for applicable inference rules.

Wilks rejects Hayes' criticism that there is no model-theoretic semantics for primitive based systems. Wilks feels that Hayes' demand for such a model-theoretic semantics makes radical Hayes' demand for a metaphysics of STUFF. Wilks emphatically rejects the application of model-theoretic semantics (in the manner of the semantics that Tarski constructed for logic) to the analysis of natural language meaning. Wilks believes that preference semantics evolves inevitably into a 'natural language' itself. However, Wilks misconstrues 'truth conditions' as serving to determine the ACTUAL truth of sentences in the object language, and gives the example of the inappropriateness of computing over a possible world. Nevertheless, possible worlds are not intended to be computational domains, but as part of an abstract conception of meaning and truth. Truth is thus only relevant to truth-determination. Model-theoretic semantics does, in fact, provide a practical means for deciding truth-determination e.g., checking whether an inference mechanism is truth-preserving.

Wilks also chides Bobrow (1975) for arguing that a primitive expansion or 'paraphrase' requires a more complex match than does the original English word that the paraphrase is for. He disputes the complexity of the matching, however, since preference semantics does not operate in paraphrase mode; he uses Schank's arguments about the paraphrase mode of Schank's primitive-based system to reject Bobrow's critique. Examining Schank's defence of primitive-based systems, we find the following list of advantages: (1) paraphrase relations are made clearer; (2) similarity relations are made clearer; (3) inferences that are true of various classes of verbs can be treated as coming from the individual (primitive) ACTs. The inferences come from ACTs and states rather than from words; and (4) organisation in memory is simplified because much information need not be duplicated. The primitive ACTs provide focal points under which information is organised.

As Schubert et al. (1979) argue, the increased clarity of paraphrase and similarity relations derives from Schank's use of canonical form rather than his basing his meaning representation on primitives. Neither can the last two advantages be traced to the use of semantic primitives. The sharing of inferences within classes of verbs can be accomplished without restating words in terms of primitive ACTs. See Cercone (1975) for the detailed example which demonstrates that both 'eats' and 'drinks' as sequential forms share in the implications that a single primitive 'ingests' would store but they are conservative of storage space and computation time. Also, argument constraints for predicates may be shared by related 'non-primitive' predicates through a constraint inheritance mechanism. Moreover, while we see no disadvantages of non-primitive based representations, point (4) shows a major disadvantage in their elimination, namely the resultant need for matching complex primitive representation instead of originally simple propositions.

The meaning formulae of primitive-based representations do single out those properties most frequently needed for comprehension and simple inferences. This is their remaining salient feature. The primitive-based representations do capture major properties of the defined concepts and we only add minor details to them. But to rely on meaning caricatures as Schank and Wilks do ensures that comprehension will remain of the crude sort. Non-primitive based representations can be equipped with the advantages of the Schank-Wilks approach, simply by providing lists of the most frequently needed properties for comprehension of each predicate and allowing the significant properties of concepts to become independently accessible without invoking the full meaning representation defining the concept. The complexity of a concept does not interfere with its matchability since it is retrieved by its name. Considerations of storage economy and the computational complexity of pattern-directed retrieval convince us of the limited value of primitive-based representations.

Having said all of the above, nevertheless, we regard the important issue centering on primitives is one of 'knowledge structuring primitives' and not an issue of defining semantic primitives. This issue is properly treated at the epistemological level of representation. We can only speculate at this time which knowledge structuring primitives should prevail and which should not.

3.3 Separating Representational and Computational Issues

Within any theory of knowledge representation systems [KRS], representational issues must be separated from computational issues. Some researchers, such as Feigenbaum (1977), treat this distinction between the theoretical and computational issues in KRS in a perfunctory manner. The terms 'knowledge engineering' and 'programs as theory' reflect this eclectic manner of thinking. However, McCarthy and Hayes (1969), Hayes (1974), and McDermott (1978) recognised the distinction and their KRS work benefited with well-defined, logically elegant representational constructs. We choose to evaluate knowledge representation systems in terms of both theoretical and computational merits, since computational performance may not reflect representational adequacy and vice-versa.

Representational issues can be investigated as they are exposed when specifying a representation language. Substituting the more neutral term 'scheme' for 'language', Hayes (1974) describes a representation scheme as a set of admissible configurations which denote domain objects and relationships between them. The interpretation assigned to a configuration depends on the scheme's semantics: the way objects in a domain correspond with representational objects, and the way configurations representing domain relationships are assigned meaning within the scheme. The former issue concerns the scheme's ontology, and determines the level of detail at which non-decomposable representation objects will serve as surrogates to 'real-world' concepts whose actual relationships are to be captured in configurations involving these surrogates. The choice of an ontology profoundly impacts the representational 'power' of a scheme. 'Block', 'pyramid', 'box', and relations like 'above', 'inside', and 'on' may capture the essence of a blocks microworld, yet the more general notions of 'set' and 'element' might be demanded by another domain. Success in interpreting configurations resulting from a domain to scheme mapping depends, to a degree, on the precision with which an ontology has been specified. Arguments for and against the use of a fixed ontology (Schank, 1974; Wilks, 1977; Schubert et al., 1979) centre on the flexibility of a representation (cf. Hayes, 1974) and the degree of detail to which interpretation is necessary for the proper treatment of a problem domain.

The second major component of a scheme's semantics facilitates interpretation of configurations involving the denoting objects, and specifies how configurations are related. This specification forms the basis for inference within a scheme, i.e., the specification of a scheme logic which can determine the implications of existing configurations. The soundess and completeness of a scheme's reasoning capabilities can be analysed only to the extent of this component's precision.

Representational and computational issues are frequently confounded when semantics are specified 'on-line' while programming. When the specification of a scheme and its associated semantics is viewed as the 'implementation' of a programming language, difficulties of interpretation are bound to arise (e.g., Bobrow and Winograd, 1977). At least some semantic theory must be prespecified and adopted as the basis for a representation scheme. The resulting 'system' should assign meanings via the chosen semantics.

Computational issues are addressed when a representation scheme is implemented. Experiments with implementation often suggest modifications to a scheme's semantics. For example, the local contexts of Conniver (McDermott and Sussman, 1974) arise from inadequacies of the single global context of Microplanner (Sussman, Winograd, and Charniak, 1970).

A knowledge representation system evolves from a knowledge representation scheme when an implementation provides the user with the facilities for managing configurations in a manner consistent with the prespecified scheme. This implementation requires the design of data structures and efficient companion algorithms which capture the specifications of the scheme. The implementation also requires the development of a human user-interface which offers the necessary capabilities without excessive, elaborate detail. The design of a representation scheme must be responsive not only to issues of representation and computation exposed upon implementation, but also to computational issues exposed along the man-machine interface. It is unlikely that a single comprehensive 'representation language' can successfully address all of the issues of knowledge representation systems.

4. Adopting the DBM Paradigm for Artificial Intelligence

4.1 DBM and AIKR: evolution and objectives

Initially DBM developed in response to the needs of large public and private organisations to organise data. The requirements of these organisations continue to dictate DBM development, and the objectives of DBM are now clearly defined (Fry and Sibley, 1976): (1) to make an integrated collection of data available to a wide variety of users; (2) to provide for quality and integrity of the data; (3) to ensure retention of privacy through security measures within the system; and (4) to allow centralised control of the data base, which is necessary for efficient data administration.

The general objectives of DBM have remained more or less unchanged; research has refined methods for achieving those objectives. In contrast, objectives for AIKR are evolving along

with the definition of AI but is instructive to formulate AIKR objectives in terms of DBM objectives.

Considering the first DBM objective, the AIKR system should provide for an intelligent program what a DBM system provides for organisational users. An AIKR system should allow manipulation, interrogation, and analysis of a repository of information which the program can consider as its 'knowledge'. Making the integrated data available begins when the AI program is defined. The AI program may be viewed as a collection of users or as a multi-faceted individual. Some AI researchers are explicit in the former, extremely modular position, e.g., Hewitt et al. (1973); Hewitt (1977).

The second DBM objective, providing for quality and integrity of the data, can be adopted directly, although in AIKR the issues pertaining to this objective are different from their manifestations in DBM. In DBM terms, the data base administrator is responsible for maintaining data quality. The human being is the conscience of the information system. The data base administrator is not explicitly incarnated in the AI paradigm, though it might be identified with that component of a system responsible for the maintenance of (say) credibility values.

Credibility or certainty factors enable a system to interpret the quality of its own data. In general, the automatic assigning of credibility values to input data also requires the maintenance of internal belief structures or user views which the system can interpret. A system which maintains some beliefs about the external world can use those beliefs to help determine the credibility of new data. MYCIN (Shortliffe, 1976) and INTERNIST (Pople, 1977) may be interpreted as attempts to automate the data quality function of a data base administrator.

DBM and AIKR have been attentive to data integrity; both argue that data integrity should be maintained automatically, DBM researchers consider a range of integrities from physical to logical, whereas AIKR has concentrated only on logical integrity. As a result of this different emphasis, DBM is more advanced in its implementation, while AIKR shows a more sophisticated treatment of data semantics. Although this gives the advantage to DBM research at the implementation level, AIKR remains more sophisticated in their treatment of data semantics.

The third objective, that of security and privacy, derives from the fear of misuse and destruction of vital data, a fear which is not yet an issue in AIKR. However, the security and privacy issue includes data security in an AIKR environment of a collection of autonomous ACTOR-like modules whose access can be both recursive and heterogenous (see Hewitt, 1977).

The final DBM objective, "to allow centralised control of the data base, which is necessary for efficient administration", is a plausible AIKR objective, with some re-interpretation. Control of the data base is centralised in the data base administrator. If the AIKR interpretation which views the data base administrator as an automatic component of the AI program is accepted, then the spirit of the objective is synonomous for both AIKR and DBM.

Whether a user views an information system as an anthropomorphic artificial intelligence, or a collection of DBM tools administrated by another human seems largely a matter of taste. However, the extent to which the data base administrative function is performed automatically does provide a criterion for distinguishing AIKR and DBM.

4.2 The DBM Methodology

A data base management system includes a data model and a data sublanguage [DSL]. The use of data sublanguages, by which a user communicates with a data base affords the DBM system independence of logical and physical representations. The notions of data model and data sublanguages are discussed with respect to the internal and external schemas commonly employed in traditional data base systems.

A data model is a strategy according to which data is organised in a data base. All data within a data base can be interpreted in the same manner, in accordance with the data model. The meaning of a collection of data is determined by the inter-relationships between the data as defined by the data model. The subtlety of data inter-relationships which a data model can capture depends primarily on the basic data structure or ontology used in the data model, e.g., popular data models include those based on relations, hierarchies, and networks. A superior data model can capture all of the meaningful relationships within the data. For the effort expended to transform the data into a form specified by the data model, a user is rewarded with domain independent techniques for manipulating his data base. These facilities depend largely on the languages provided for the use of a data model.

Data sublanguages including data definition languages [DDL] and data manipulation languages [DML] facilitate communication with a data base. A DDL is used to describe the relationships within some data in terms consistent with the structural constraints imposed by the data model. A user must describe the relations within his data in terms of the concepts provided by the DDL, which derive from the data model. The use of a consistent DDL permits maintenance and manipulation of the data base by domain independent software and hardware. A DML uses the definitions provided in the DDL to enable the user to update, insert, and retrieve data rendered in terms of the data model.

Some DSL's, such as those which consist of DBM primitive operations embedded in a general purpose programming language, keep the user in close contact with the manipulation of his data. These DSL's manipulate data in terms of the low-

est level data relationships or structures, i.e., the basic components of the data model. Other DSL's provide a complete complement of computational functions independent of any other system. These more sophisticated systems for manipulating data do so in terms of the data descriptions. For example, data input and verification may be entirely the user's responsiblity or data can be automatically collected and verified with the user supplied data definitions. The level of sophistication required in a DSL is a function of the data model, the user, and the user's data. DSL's will continue to develop in response to new combinations of those factors. The notion of external schema (or schemata) is closely related to the notion of DDL.

The interface between the user's view of his data and the DBM system's view of his data is defined by the data model is an important interface. The external schema is the set of DDL descriptions which comprise a user's view of his data. The external schema enables the user to conceptualise his data in terms of the data model. The user first conceives his data inter-relationships in his external schema, imposing the data model on his data by means of user written DDL expressions. External schemas may be unique to a given user's data base, or several external schemas describing all or portions of a single data base might provide multiple user viewpoints.

The internal schema is the DBM system component which manages data at the level of physical devices. The internal schema imposes the data model on the internal representation of the data. A user conceptualises his data in an external schema, which is formalised in a data model, and the data model is implemented via the internal schema of the DBM system. At the internal schema level, user's expressions in a data manipulation language are finally interpreted. At this level, the abstract data model is optimised for storage, retrieval, and modification of user data. The fidelity with which the internal schema imposes the data model onto the physical devices which will perform storage, retrieval, and modification of the data is the most important factor in the overall performance of the DBM system. The internal schema implements the data model in a computationally efficient manner, and insulates the user from all physical data management considerations. A DBM system which provides users this insulation provides 'data independence', the greatest achievement of good data management.

4.3 Applying DBM to Artificial Intelligence

Having introduced DBM systems, we speculate on the applicability of DBM notions to AI representation systems. Currently DBM has better defined and more comprehensive ideas of data management than AI, especially at the level of physical representation and manipulation of data on actual devices and DBM systems architecture provides a structured framework in which to view AI representation schemes.

Comprehensive knowledge representation languages [KRL] (e.g., FRL, Roberts and Goldstein, 1977; KRL, Bobrow and Winograd, 1977; NETL, Fahlman, 1979; KL-ONE, Brachman, 1978) are AI's attempts to develop an AI data base management system. Knowledge representation languages act as the data sublanguages of DBM systems. DSL's facilitate communication with a data base and KRL's perform a similar function for AI data bases. KRL's are more ambitious in design because they attempt to accomplish all of the functions of DBM with a single programming system.

A comprehensive symbol manipulation includes the data definition language and data manipulation language of DBM systems. It is a disadvantage that aspects of data representation and manipulation which are clearly separable in DBM become confused in the monolithic KRL approach. The concept of a data model becomes less clear, especially when data representations overlap with procedures for their manipulation. KRL-0's (Bobrow and Winograd, 1977) procedural attachment demonstrates this dissolution of data management and encourages an implementational treatment of data model specification. KRL designers generally specify data models, but rather imprecisely, so that the data model is susceptible to adjustment of its semantics upon computer implementation. Once the data model becomes diffused, it is difficult to distinguish the data base of representing expressions from the rest of the system. If the representing expressions are identified, it is still difficult to interpret them consistently and uniformly since the semantics of the data model are rendered only procedurally, by implementation.

Frames, scripts, and schemata organise the knowledge which they represent according to the function of that knowledge. The following example illustrates the functional organisation for frames.

> Knowledge associatively accessed via frame slots can be procedurally embedded; execution of these procedures provides a computational comprehension for concept and context. In the sentences (see Schubert et al., 1979)
>
> John unlocked his car. (1)
> He used his key. (2)
>
> John graded the exam. (3)
> He used his key. (4)
>
> the meaning of 'key' changes from sentence (2) to sentence (4), comprehension of its different meanings depends upon the 'unlockingcars' frame and the 'gradingexams' frame. The frames might contain slots for carkeys and examinationkeys, or slots for other frames to explain car and examination keys. The frames might also contain information about the kind and use of keys in general. The description of 'his key' guides clever access mechanisms to select the appropriate slot.

This functional organisation of frames, scripts, and schemata makes them the AI counter-

part to the DBM data model. They serve as an epistemological level representation (cf. Brachman, 1979) between the AI 'external schema' (conceptual level representation) and the AI 'internal schema' (logical level and implementation level representation). Thus, functionally organised representations can be evaluated as data models. The use of frames (scripts, schemata) as a data model should free the knowledge representation system from external and internal schema considerations until the semantics of the proposed model are investigated. Unfortunately, the semantics of these models has not been completely investigated. As a result, schema considerations have been ignored, or considered prematurely. The semantics of functionally organised representations are frequently treated as a computational rather than a representational issue. To be effective 'procedural semantics' (e.g., Winograd, 1976) must distinguish between implementing a well-specified notion and specifying a notion by implementing it. For example, various frame based systems treat the selection of replacement frames differently, as a computational problem to be solved by clever implementation, rather than as a representational issue to be solved with well-specified semantics which delineate various selection classes.

Associative networks (including semantic networks, see Findler, 1979) are organised according to the concepts which they represent. This concept-centred organisation is the definitive characteristic of associative networks.

Associative networks correspond closely to the DBM internal schema because networks provide an accessing structure for the concepts which they represent. Associative networks have also been used as the basic data model for AI knowledge representation systems, usually as alternatives to frames, scripts, and schemata. The functional organisation common to frames, scripts, and schemata make them the AI counterpart to the DBM model. However, since networks provide an accessing structure for the concepts they represent, associative networks correspond more closely to the DBM internal schema than frames, scripts, and schemata. The network organisation operates as an index for the concepts, their properties and descriptions, this organisation is closely related to the DBM internal schema notion. All of the knowledge about a particular concept is accessible from that concept. In a similar manner, the internal schema instantiates the data model as the actual representation of data in a data base. In conventional DBM, the data model defined all items of data as interrelated, so the internal schema was the actual relationship between items of data, which made them accessible from any single data item.

Even in non-network representations, such as KRL (Bobrow and Winograd, 1977) and PLANNER (Hewitt, 1972) network organisations are used in implementation. The concept or 'object' centred organisation is a natural choice for an internal indexing structure. This is due, in part, to the earlier psychological notions about the organisation of memory (see Anderson and Bower, 1973; Wilson, 1979). In the process of implementation, when the equivalent of an internal schema must be defined, network organisations are used. This not only strengthens the parallel between associative networks and logical organisations but also indicates that network organisations are useful ones.

Network proponents who draw parallels between conceptual organisations and external schemas imply that the conceptual organisation parallels the user's conception of his data expressed in terms of the data model. To allow conceptual organisations to be identified with an external schema, these researchers sacrifice the independence between internal and external schemas. This violates a fundamental DBM tenent of external and internal schemas. External and internal schemas were originally distinguished by DBM in order to achieve data independence. It would be contrary to a major objective of DBM, that is data independence, to permit amalgamation of the internal and external schemas.

A production system includes a set of rules (productions), a global data base, and an interpreter for evaluating rules in terms of the data base (Davis and King, 1977). The production system data base typically stores state information about a domain, and production rules specify how state changes in the data base can be effected. The interpreter acts as a kind of pattern matcher, selecting rules according to some a priori rule ordering, then applying them to the data base to possibly affect subsequent rule applications (cf. Microplanner, Sussman et al., 1971). Production systems differ from frames, scripts, schemata, and networks in that they include methods for changing the information stored. The multiple component nature of production systems precludes a direct correspondence with any single part of the DBM paradigm. Their data base component gives production systems the appearance of a (naive) DBM scheme which, with a suitable interface, would provide a comprehensive information management tool.

However, productions systems data bases lack a unique data model (Davis and King, 1977). The structure of each data base and its production rules is determined by the application (e.g., DENDRAL, Feigenbaum et al., 1971). For example, rule patterns and data base expressions of DENDRAL are graphs representing chemical structures. The data model is motivated by the problem domain, where the structure of molecular components is to be determined from mass spectogram data. Alternatively, MYCIN (Shortliffe, 1976) uses a data model based on diagnostic rules which, when applied to expressions representing medical laboratory test results, produces a numeric measure of credibility for a particular diagnosis. Each rule is assigned an a priori credibility which contributes to the credibility of a suggested diagnosis. Both systems rival human performance in many cases. With or without data models, production systems manage data bases in a more structured manner than do most AI systems, and provide a framework in which to explore issues of representation.

4.4 A Data Model for Artificial Intelligence

Data models are specified in response to user expectations of his data domain. Conventional

DBM data models were limited to a simple extensional treatment of the domain partly because such a treatment seemed sufficient for the data concerned, and partly because data models which would capture intensional knowledge would delay implementation of the data base (Wong and Mylopolous, 1977). Artificial intelligence domains are expected to accommodate more complex 'human-like' use of data, therefore AI developed more sophisticated representation schemes to represent intensional knowledge. Representation ideas which are immature with respect to what AI researcher require of domains greatly extend the domain with respect to the expectations of a conventional data base. DBM researchers are becoming responsive to the advantages accruing from more sophisticated data models (Codd, 1979).

If AI specified a data model, it would separate internal and external schemas, and foster investigations of data independence. Fidelity to an AI data model would eliminate ad-hoc representations. Like their DBM counterparts, representation theorists would be forced to separate representational and implementational issues. Without this separation the notions of internal and external schemas and data independence would lose their meaning.

Currently more important, AI representation schemes could be comparatively evaluated with respect to one another if each one declared its data model. There exists a surfeit of representational schemes, each raising a host of questions and issues. A unified data model would identify truly general representational issues rather than becoming restricted to the problems of one particular representation.

A precisely specified data model must define exactly how each data base expression corresponds with the problem domain. Since logical semantics are precise, we suggest logic as a data model. The logic will bring semantic consistency to the data model. Arbitrarily complex logical sentences have repeatedly been interpreted using logical semantics and the interpretations have been evaluated. The interpretations have been precise enough to forestall criticism of the logic contained in them (Minsky, 1975). Current approaches to automatic programming, data base management, and problem solving indicate that the logical perspective has again become fashionable (van Emden, 1977; Gallaire and Minker, 1978; McDermott and Doyle, 1978). Those as yet unconvinced that the logical point of view has gained ascendency should consult Hayes (1977).

Logical data models have been used by researchers who have not yet explicitly defined a data model. Schubert (1975) relied on a logical data model to give precise interpretations to semantic networks. Schubert's work demonstrates how expressions from a logical data model can be extended to a structure which provides an access scheme for represented knowledge. The logical data model demonstrates how the associative network functions as an internal schema. The logical data model has elucidated interpretations for representations with more vague data models, such as frames. Hayes (1977b) discusses 'the logic of frames' and offers logical interpretations for the structure of frames. In his logical analysis he finds that frames have the potential for reflexive reasoning (i.e., a frame knowing something about itself). The logical data model provided the tools necessary for such an evaluation.

Using a logical data model, Hayes demonstrates how the syntax of representation languages can obscure the underlying data model. Syntaxes which are designed to provide amenable user-level languages frequently disguise the underlying data model. This is another instance of confusion between the human-engineered interface and the data model, which the precise specification of the data model can resolve.

As is the case with traditional DBM data models, the requirements anticipated by potential users dictate the form of a logic-based data model. An AI data model should provide a sophisticated treatment of representation. However, since programming invariably uncovers unforeseen difficulties, pressure for experimental implementations affect the specification of refinements in successive data models.

A logic-based data model will consist of a logical language whose semantics offer a precise interpretation for each language expression. A standard syntax will suffice since any difficulty with the syntax of data model expressions should be alleviated by a suitable user DSL, as with the relational DBM model. The experience of the KRL-0 design group would attest to this point.

We expect the logic used as the basis for a data model to include proof theory which can be used to manipulate expressions of the data model. The capabilities of the proof theory will depend on the logical system adopted and how expressions of the corresponding data model are interpreted with respect to a domain (Nicolas and Gallaire, 1977).

The use of logic to describe the data considered by traditional DBM data models illuminates the development of a logic-based AI data model. An edited volume (Gallaire and Minker, 1977) treats this subject exclusively. One current approach (Kowalski, 1977) renders the extensional relations of a relational model as base relations in a first-order logic. Kowalski discusses how general statements (i.e., quantified formulae) can be interpreted procedurally to maintain integrity or derive implications in the extensional data base. Two levels of representation can be identified: the base relations which express a collection of extensional facts, and a collection of general expressions which express knowledge about the base relations. We concur with Kowalski that a more sophisticated approach would consider the general expressions as data given in the data model.

Recent representation investigations (Brachman, 1979; Schubert et al., 1979; Davis, 1976) suggest the use of several levels of knowledge representation. Each level would express information about the lower level with the base

level corresponding to an actual domain interpreted via a traditional logical semantics (e.g., Tarskian). Nonetheless, it would seem that a more elaborate data model is necessary, for example, a model in which sentences in a meta-language express knowledge about how to manipulate the sentences in a corresponding object language. The GOLUX project described briefly by Hayes (1974) is concerned with the specification of such a language. Meta-level information, when interpreted, provides control information for the use of object-level information. Whether the implied regress can be closed within a single comprehensive formal system remains an open question.

Reiter's (1979) default logic provides a powerful system for viewing a data base of first-order and default expressions as a consistent set of beliefs about a domain. A default is a statement which suggests new information, which is assumed true, will not be inconsistent with the existing data base. The logic is non-monotonic, i.e., new axioms can invalidate old theorems. The non-monotonic logic of McDermott and Doyle (1978) is another candidate for consideration. Since the formal interpretaion of meta-level expressions requires a higher level logic, the second order theory of Gilmore (1971) can also be investigated as a possible data model.

5. Final Remarks

We conclude by urging the adoption of the DBM paradigm by AI researchers and the development of a strong data model of AI. The success demonstrated by AI in the retrieval of information implicit in the data (knowledge) base and manifested by the various pattern-directed (procedure-invoked) matching algorithms should contribute significantly to the development of semantics for the data model. Nevertheless, the concept of a data model must first be incorporated by AI into their systems. The adoption of the DBM data model should prove to be an invaluable vehicle for improving the performance of AI systems which operate with a large knowledge base.

Acknowledgements

We wish to thank Carol Murchison and Josie Backhouse for editing earlier drafts of this paper and making invaluable suggestions. This research was supported by the National Science and Engineering Research Council of Canada under Operating Grant no. A4309 and by the Office of the Academic Vice-President, Simon Fraser University.

References

Anderson, J. and Bower, G. (1973). HUMAN ASSOCIATIVE MEMORY, New York, N.Y., Holt Reinhart.

Black, F. (1968). "A Deductive Question-Answering System", SEMANTIC INFORMATION PROCESSING. (ed M. Minsky) MIT Press, Cambridge, Mass., 354-402.

Bobrow, D. & Winograd, T. (1977) "An overview of KRL, a Knowledge Representation Language", Cognitive Science 1, 3-46.

Brachman, R. (1979). "On the Epistemological Status of Semantic Networks", ASSOCIATIVE NETWORKS: Representation and Use of Knowledge by Computers. (ed N. Findler) Academic Press, New York, 3-46.

Brachman, R. (1978). "Theoretical Studies in Natural Language Understanding", Report No. 3888, BBN Inc., Cambridge, Mass.

Cercone, N. (1975). "Representing Natural Language in Extended Semantic Networks", PhD Thesis, Tech. Report TR75-11, Dept. of Computing Science, Univ. of Alberta, Edmonton, Alberta.

Chamberlin, D. (1976). "Relational Data Base Management Systems", ACM Computing Surveys 8, 43-66.

Codd, E. (1979). "Extending the Data Base Relational Model", Supplement to Proceedings of ACM SIGMOD 1979, 29-52.

Davis, R. (1976). "Applications of Meta-level Knowledge to the Construction, Maintenance and Use of Large Knowledge Bases", Stanford AI Lab Memo AIM-283, Stanford, Cal.

Davis, R., Buchanan, B., and Shortliffe, E. (1975). "Production Rules as a Representation for a Knowledge Based Consultation Systems", Artificial Intelligence 8, 15-45.

Davis, R. & King, J. (1977). "An Overview of Production Systems", MACHINE INTELLIGENCE 8, (eds) Elcock, E. & Michie, D., Ellis Horwood Ltd., 300-332.

Doyle, J. (1978). "Truth Maintenance Systems for Problem Solving", AI Lab Memo TR-419, MIT, Cambridge, Mass.

Fahlman, S. (1979). NETL: A System for Representing and Using Real World Knowledge, MIT Press, Cambridge, Mass.

Feigenbaum, E. (1977). "The Art of Artificial Intelligence: Themes and Case Studies of Knowledge Engineering", Proceeding of IJCAI5, Cambridge, Mass., 1014-1029.

Feigenbaum, E., Buchanan, B. & Lederberg, J. (1971). "On Generality and Problem Solving: A Case Study using the DENDRAL Program", MACHINE INTELLIGENCE 6, (eds) Meltzer, B. & Michie, D., American Elsevier, New York, 165-189.

Findler, N. (1979). ASSOCIATIVE NETWORKS: Representation and Use of Knowledge by Computers, Academic Press, New York.

Fry, J. & Sibley, E. (1976). "Evolution of Data-Base Management Systems", ACM Computing Surveys 8, 7-42.

Gallaire, H. & Minker, J. (1978). LOGIC AND DATA BASES, Plenum Press, New York.

Gilmore, P. (1971). "A Consistent Naive Set Theory: Foundations for a Formal Theory of Computation", IBM Research Report RC 3413, Yorktown Heights, New York.

Goebel, R. (1977). "Organising Factual Knowledge in a Semantic Network", TR77-8, Dept. of Computing Science, Univ. of Alberta, Edmonton, Alta.

Green, L., Berkeley, E & Gotlieb, C. (1959). "Conversation with a Computer", Computers and Automation 8, 9-11

Hayes, P. (1977). "In Defence of Logic", Proc. of IJCAI5, Cambridge, Mass., 559-565

Hayes, P. (1977b). "The Logic of Frames", Dept. of Computer Science, Univ. of Essex.

Hayes, P. (1974). "Some Issues and Non-Issues in Representation Theory", Proceedings of the AISB Summer Conference, Univ. of Sussex, Brighton , 63-79.

Hendrix, G. (1975). "Expanding the Utility of Semantic Networks Through Partitioning", Proc. of IJCAI4, Tbilisi, USSR 115-121.

Hewitt, C. (1977). "Viewing Control Structures as

Patterns of Passing Messages", Artificial Intelligence 8, 323-364.
Hewitt, C. (1972). "Description and Theoretical Analysis (using schemata) of PLANNER: A Language for Proving Theorems and Manipulating Models in a Robot", AI Lab Memo TR-258, MIT, Cambridge, Mass.
Hewitt, C., Bishop, P. & Steiger, R. (1973). "A Universal ACTOR Formalism for Artificial Intelligence", Proc. of IJCAI3, Stanford, Cal., 235-245.
Kowalski, R. (1977). "Logic for Data Description", LOGIC AND DATA BASES, (eds) Gallaire, H. & Minker, J., Plenum Press, New York, 77-103.
Lindsay, R. (1973). "In Defence of Ad-Hoc Systems" COMPUTER MODELS OF THOUGHT AND LANGUAGE (eds) Schank, R. & Colby, K., Freeman, San Fran., 372-395.
Lindsay, R. (1963). "Inferential Memory as the Basis of Machines Which Understand Natural Language", COMPUTERS AND THOUGHT. (eds) Feigenbaum, E. & Feldman, J., McGraw Hill, New York, 217-233.
McCarthy, J. & Hayes, P. (1969). "Some Philosophical Problems from the Standpoint of Artificial Intelligence", MACHINE INTELLIGENCE 4, (eds) Meltzer, B. & Michie, D., American Elsevier, New York, 463-502.
McDermott, D. (1978). "The Last Survey of Knowledge Representation", Proc. of the AISB/GI Conf. on AI, Hamburg, Germany, 206-221.
McDermott, D. (1978b). "Tarskian Semantics, or No Notation Without Denotation", Cognitive Science 2, 277-282.
McDermott, D. & Doyle, J. (1979). "Non Monotonic Logic I", AI Lab Memo 489A, MIT, Cambridge, Mass.
McDermott, D. & Sussman, G. (1974). "The Conniver Reference Manual", AI Lab Memo 259A, MIT, Cambridge, Mass.
Minsky, M. (1975). "A Framework for Representing Knowledge", THE PSYCHOLOGY OF COMPUTER VISION. (ed) P. Winston, McGraw Hill, New York, 211-277.
Moore, J. & Newell, A. (1973). "How Can Merlin Understand", Dept. of Computer Science, Carnegie-Mellon Univ., Pittsburgh.
Newell, A. (1973). "Production Systems: Models of Control Structures", VISUAL INFORMATION PROCESSING. (ed) W. Chase, Academic Press, New York, 463-526.
Nicolas, J. & Gallaire, H. (1978). "Data Bases: Theory vs. Interpretation", LOGIC AND DATA BASES. (eds) Gallaire, H. & Minker, J., Plenum Press, N.Y. 33-54.
Pople, H. (1977). "The Formation of Hypothesis in Diagnotic Problem Solving, and Exercise in Synthetic Reasoning", Proceedings IJCAI5, Cambridge, Mass., 1030-1037.
Quillian, M. (1968). "Semantic Memory", SEMANTIC INFORMATION PROCESSING. (ed) M. Minsky, MIT Press, Cambridge, Mass. 227-270.
Raphael, B. (1968). "SIR: A Computer Program for Semantic Information Retrieval", SEMANTIC INFORMATION PROCESSING. (ed) M. Minsky, MIT Press, Cambridge, Mass. 33-145.
Reiter, R. (1979). "A Logic for Default Reasoning", Artificial Intelligence
Roberts, R. & Goldstein, I. (1977). "The FRL Manual", AI Lab Memo 409, MIT, Cambridge, Mass.
Rumelhart, D., Lindsay, P. & Norman, D. (1972). "A Process Model for Long Term Memory", ORGANISATION OF MEMORY. (eds) Tulving, E. & Donaldson, W., Academic Press, New York, 197-246.
Schank, R. (1975). "The Role of Memory in Language Processing", THE STRUCTURE OF HUMAN MEMORY. (ed) C. Cofer, Freeman, San Francisco, 162-189.
Schank, R. (1972). "Conceptual Dependency: A Theory of Natural Language Understanding", Cognitive Psychology 3, 552-631.
Schank, R., Goldman, N., Rieger, C. & Riesbeck,C. (1973). "MARGIE: Memory Analysis, Response Generation and Inference in English", Proc. of IJCAI3, 255-261.
Schubert, L. (1976). "Extending the Expressive Power of Semantic Networks", Artificial Intelligence 7, 163-198.
Schubert, L., Goebel, R. & Cercone, N. (1979). "The Structure and Organisation of a Semantic Net for Comprehension and Inference", ASSOCIATIVE NETWORKS: The Representation and Use of Knowledge by Computers. (ed) N. Findler, Academic Press, New York, 121-175.
Shapiro, S. (1971). "A Net Structure for Semantic Storage, Deduction, and Retrieval", Proc of IJCAI2, London, 512-523.
Shortliffe, E. (1976). COMPUTER BASED MEDICAL CONSULTATIONS: MYCIN. American Elseview, NY.
Simmons, R., Klein, S. & McConologue, K. (1964). "Maximum Depth Indexing for Computer Retrieval of English Language Data", American Documentation 15, 196-204.
Smith, B. & Brachman, R. (1977). "How is a Knowledge Representation System Like a Piano", AI Lab Memo 497, MIT, Cambridge, Mass.
Sussman, G., Winograd, T. & Charniak, E. (1971). "Micro-Planner Reference Manual", AI Lab Memo 203A, MIT, Cambridge, Mass.
Szolovits, P., Hawkison, L. & Martin, W. (1977). "An Overview or OWL: A Language for Knowledge Representation", MIT/LCS/TM-86, MIT, Cambridge, Mass.
Van Emden, M. (1977). "Programming with Resolution Logic", MACHINE INTELLIGENCE 8, (eds) Elcock, E. & Michie, D., Ellis Horwood Ltd. 266-299.
Waterman, D. (1970). "Generalised Learning Techniques for Automating the Learning of Heuristics", Artificial Intelligence 1, 121-170.
Wilks, Y. (1977). "Good and Bad Arguments about Semantic Primitives", D.A.I. Research Report No. 42, Univ. of Edinburgh, Edinburgh.
Wilks, Y. (1973). "Preference Semantics", Stanford AI Project, memo AIM-206, Stanford Univ., Stanford, Cal.
Wilson, K. (1979). FROM ASSOCIATION TO STRUCTURE. North Holland, Amsterdam.
Winograd, T. (1976). "Towards a Procedural Understanding of Semantics", AI Lab Memo AIM 292, Stanford Univ., Stanford, Cal.
Winograd, T. (1972). UNDERSTANDING NATURAL LANGUAGE. Academic Press, New York.
Wong, H. & Mylopolous, J. (1977). "Two Views of Data Semantics: A Survey of Data Models in Artificial Intelligence and Data Base Management", Infor 15, 344-383.
Woods, W. (1975). "What's in a Link", REPRESENTATION AND UNDERSTANDING, (eds) Bobrow, D. & Collins, A., Academic Press, New York, 35-82.
Woods, W., Kaplan, R. & Nash-Webber, B. (1972). "The Lunar Sciences Natural Language Information System: Final Report", Bolt, Beranek and Newman, Inc., Cambridge, Mass.

DIFFERENCES AND SIMILARITIES BETWEEN THE TWO INFERENCE MODES OF THE *RESEDA* SYSTEM, THE "HYPOTHESIS" MODE AND THE "TRANSFORMATION" MODE.

Gian Piero ZARRI

Centre National de la Recherche Scientifique
Laboratoire d'Informatique pour les Sciences de l'Homme
54, Boulevard Raspail - 75270 PARIS CEDEX 06 FRANCE

ABSTRACT

The principal characteristic of the RESEDA system is being able to question a biographical database about the causal relationships which may be inferred between different attested facts in the base. This paper deals with the two classes of inference procedures existing in the system, "hypotheses" and "transformations", and their complementary practical aims. It will show that, from a very general point of view, the differences between the two classes only concern the instrumental level : in theory, a same common sense rule can be interpreted, and executed, equally well as a "hypothesis" or as a "transformation".

INTRODUCTION

RESEDA* is a system for managing a biographical database using Artificial Intelligence (AI) techniques. The term "biographical data" must be understood in its widest possible meaning : being in fact any event, in the public or private life, physical or intellectual, etc., that it is possible to gather about the personages we are interested in. In the present state of the system, this information concerns a well-defined period in time (approximately between 1350 and 1450) and a particular subject area (French history), but the techniques used in RESEDA could just as well be applied to biographical information concerning other areas (medicine, law, etc.).

RESEDA differs from "classical" factual database management systems in two ways :

a) The information is recorded in the base using a particular Data Definition Language (metalanguage) which uses knowledge representation techniques.

b) A user interrogating the base obtains not only information which was directly introduced into it but also implicit information found using *inference mechanisms* particular to the system - see, in the same vein, Carbonell (1978), Kolodner (1980 ; 1981), McGregor and Malone (1981), etc.

In this article, I shall deal mainly with the aspects of the system concerning the "inference procedures". More specifically, after having explained the differing practical aims of the two ways of using the inferences, "hypothesis" mode and "transformation" mode, I shall try to show that, from a very general point of view, that difference only concerns the instrumental level : in theory, the same common sense rule can be interpreted, and executed, equally well as a "hypothesis" or as a "transformation". I shall conclude by noting that, in practice, the kind of searches that RESEDA favours actually impede systematic switching between one interpretation mode and the other. Knowledge of their "deep" identity does however enable a more coherent systematization, on a conceptual level as on the level of practical efficiency, of RESEDA's inference mechanisms.

THE *RESEDA* SYSTEM AND ITS INFERENCE RULES

I shall first of all state some fundamental facts about RESEDA.

The biographical information which constitutes the system's database is organized in the form of units called "coded episodes" or "planes". There are several different types of plane ; the "predicative plane", the most important, corresponds to a "flash" which illustrates a particular moment in the "life story" of one or more persons. A predicative plane is made up of one of five possible "predicates" (BE-AFFECTED-BY, BEHAVE, BE-PRESENT, MOVE, PRODUCE), to which one or more "modulators" may be attached. The modulator's function is to specify and delimit the semantic role of the predicate. Of course, the "meaning" of the modulator plus predicate is "defined" - as for all elements of the RESEDA Data Definition Language - by the general behaviour of the system rather than by the usual function of these codes in natural language.

---

* The RESEDA project is financed by grants from the "Délégation Générale à la Recherche Scientifique et Technique" (RESEDA/0, CNRS-DGRST Contract n° 75.7.0456), from the "Institut de Recherche d'Informatique et d'Automatique" (RESEDA/1, CNRS-IRIA Contract n° 78.206), and from the "Centre National de la Recherche Scientifique" within the framework of the "Action Thématique Programmée Intelligence Artificielle".

The predicate of the plane is accompanied by "case slots" (Rosner and Somers 1980 ; Charniak 1981) which introduce the predicate arguments.

Dating and space location information is also given within a predicative plane, as is the bibliographic authority for the statement. Predicative planes may be linked either through the label of one plane being the value of an argument slot (the slot OBJ) in another, or through explicit links "and", "or", "cause", "finality", etc.

The extremely simple example given in figure 1 should provide a clearer idea of what I have just explained. It is the representation of "Montreuil was opposed, between 1413 and 1416, to the Burgundian party on the subject of the Hundrer Years War"; the bibliographical authority is the historian Valois. The codes given in capital letters indicate

```
1)  against+BEHAVE   SUBJ   Montreuil : Paris
                     OBJ    burgundians
                     ARG    hundred-years-war
                     date1  : 1413
                     date2  : 1416
                     bibl   : Valois,IV
```

figure 1

the predicate and the cases associated with it ; "against" is a modulator. For each predicative plane there is a pair of temporal markers, "date1-date2", which give the duration of the episode. "Montreuil" is one of the historical personages whose "life story" is recorded in the database ; "burgundians" and "hundred-years-war" are entries in RESEDA's lexicon, which provides the historical background of the system. "Paris" is obviously the "subject location". If the historical documents give us some explanation for the Montreuil's attitude recorded in plane 1, then the corresponding planes would be introduced into the database and the plane 1 would be associated with them by an explicit link of the "cause" type.

When the system is considered from the point of view of its utilization, the fundamental concept which must be introduced is that of the "search model". A "search model" gives the essential elements, expressed in terms of the RESEDA metalanguage, of a coded episode which it is necessary to search for in the database. A search model may originate from outside the system, if it is the direct translation of a query posed by a user. On the other hand, it may be automatically generated by the system, as will be clarified later on, during the execution of an inference procedure.

Let us suppose, for example, that a user is interrogating the system about the relationships between Montreuil and the Burgundian party concerning the Hundred Years War. In this case, the user himself creates the search model given in figure 2, with the aid of a prompting program. The only notable difference between this formalism and the formalism required for the representation of the episodes in the database is that of the presence of a "search interval", "bound1-bound2", which

```
BEHAVE   SUBJ   Montreuil
         OBJ    burgundians
         ARG    hundred-years-war
         bound1 : 1400
         bound2 : 1420
```

figure 2

the user will employ to limit the period he considers appropriate to explore according to the information for which he is searching.

I do not intend, here, to go into the details of the procedure adopted to test the match of a search model with data in the base ; instead, for details of this, see Zarri *et al.* (1979 ; 1980). It is, for example, obvious that the model in figure 2 may be directly matched with the plane in figure 1 ; this of course is the exception rather than the rule.

In the case of a dead end, a first class of inference rules may be applied to the model, that is the "transformations". To keep to an extremely simple example, the search model : "(for+BEHAVE SUBJ *x* OBJ *y*) = to be favourable towards ...", could be substituted by the model : "(against+BEHAVE SUBJ *x* OBJ *y*)", given that information regarding the unfavourable attitude of person *x* toward person *y* is at the same time a response to any query about the possibility of a favourable attitude. Note the existence of an underlying common sense rule even in such a simple transformation.

A second example of a transformation is that given in figure 3. The underlying common sense rule is : "if a person *x* has a university degree *w*, then this person has followed some course *v*" (one or several persons *y* have "produced" the course *v* with the intention of *x*). In figure 3, I have only partially detailed the "restrictions" associated withe the "variables" *x*, *y*, *v* and *w*; the use of variables allows maximum generality in the formulation of

```
PRODUCE   SUBJ   y  ⇨  BE-AFFECTED-BY   SUBJ  x
          OBJ    v                      OBJ   w
          DEST   x

x = <personage>
y = <personage>|<personages>
x ≠ y
v = <university-course>
w = <degree-obtained>
w = f(v)
```

figure 3

the common sense law underlying the transformation. The values which replace the variables in the retrieved plane (or planes) using the transformed model must obviously respect these

restrictions ; for example, the particular type of "degree" which will have been substituted for *w* must be compatible with the value of *v* in the original model, which I have indicated for simplicity as *w = f(v)*.

Unlike the transformation "for/against" given earlier, the transformation in figure 3 is "one way", that is, it is possible to transform the model from left to right, but not the inverse.

Very often the passage of one model to another is subject to certain conditions ; it is only possible to substitute one model for the other if a particular condition has been satisfied. This is verified by checking the existence of episodes within the base which are able to guarantee the appropriate context ; examples of conditional transformations will be examined in the next section.

Note that the use of concepts comparable with our "semantic transformations" is quite common when using AI techniques to exploit a factual database, see for example "elaborations" in Kolodner (1980), "extensions" in Hafner (1981), "expansions" in McCarty and Sridharan (1980), etc.

Even taking into consideration this first category of inference rules, the behaviour of the system such as it has been described up to now is entirely classic in type. There is, however, a second, more original way of searching RESEDA : it is possible to search for the implicit "causes", in the widest sens of the word, of an attested fact in the base. For example, if the user, in submitting the query in figure 2, obtained in reply the plane in figure 1 and if we assume that the "causes" of this negative attitude of Montreuil towards the Burgundians are not explicitly recorded in the database, he will now be able to ask the system to automatically produce a *plausible* explanation of this attitude by using a second category of inference rules, the "hypotheses".

In order to give a first idea, on an intuitive level, of the functioning of the hypotheses, figure 4 shows the formulation in natural language of four characteristic hypotheses of the RESEDA system.

The first part of each of these rules corresponds to a particular class of confirmed facts (planes) for which one asks the causes. For example, the plane in figure 1 is clearly an exemplification of the first part of the third hypothesis in figure 4. In RESEDA's terminology, the formal drafting of the first part is called a "premiss". The second part (the "condition") gives instructions for searching the database for information which would be able to justify the fact which has been matched with the premiss. That is, if planes matching the particular search model which can be obtained from the "condition" part of the hypothesis can be found in the database, it is considered that the facts represented by these planes *could* constitute a justification for the plan-premiss and are then returned as the response to the user's query. An important point to notice is that search models generated by the hypotheses, like any search model, can be transformed as well in the case of an initial failure of the match procedures.

a) ... one might cease to act on behalf of some other person

BECAUSE

one has abused that person's confidence (e.g. by misrepresenting his views to a third party)

b) ... one might leave something (in one's will) to a (religious) community

BECAUSE

one had some special connection with this community

c) ... one might take a particular attitude in an argument

BECAUSE

one has close links with one of the parties in a conflicting situation

d) ... one might lose a position (of civil servant)

BECAUSE

one is replaced by a supporter of the (political) party which has just taken power.

figure 4

In the case of the enquiry about the causes of Montreuil's attitude, the search models generated by the hypothesis in figure 4c enable the retrieval from the biographical base of the planes in figure 5 ; plane 2 translates the fact that Montreuil (and Gontier Col) were amongst the Armagnac ranks between 1400 and 1415, while plane 3 tells us that the Armagnacs and Burgundians were in conflict between 1407 and 1425.

The following is a "résumé" of what I have expressed in the preceding paragraphs. There exist in RESEDA two fundamental ways of retrieving information requested by a user. In the first

```
2) BE-AFFECTED-BY   SUBJ   armagnacs
                    OBJ    (COORD1 Montreuil Col)
                    date1  : 1400
                    date2  : 1415
                    bibl   : Le-Duc

3) recip+against+BEHAVE   SUBJ  (COORD1 armagnacs
                                 burgundians) :
                                 France
                          OBJ   (COORD1 armagnacs
                                 burgundians) :
                                 France
                          date1 : 1407
                          date2 : 1425
                          bibl  : consensus
```

figure 5

case, the information which we wish to obtain is data that already exists in the base and is

retrieved as such, without further processing. This data can be obtained by direct match of the search model that corresponds to the user's question. If this is not possible , we can still try to get an answer by using the inference procedures of the "transformation" type. The second method retrieves information which, on the contrary and most importantly, is created *ex nihilo* by the search procedure itself. It expresses, in fact, the possibility of a new causal relationship, in the base, between the coded episode provided explicitly by the user and one or more planes that the system retrieves by applying the "condition" part of an hypothesis. In this second approach, and to the opposite of what happens in the first, the research procedure modifies in some way the primitive distribution of the data. The second method is only possible by directly employing a particular category of inference procedures, those of the type "hypothesis" : transformations can still intervene in this framework, however, to aid the matching of the search models obtained from the condition.

THE SUBSTRATUM COMMON TO *RESEDA*'S INFERENCE OPERATIONS

Let us now look in some detail at the hypothesis shown in figure 4d. A whole family of inference rules expressed in RESEDA's metalanguage corresponds in reality to the natural language formulation shown in this figure ; one of these realisations is shown in figure 6. A description of the procedure followed in order to isolate the elements of these families can be found in Zarri (1981) ; see also Zarri (1979) for the general methodology for constructing hypotheses.

The formulation in figure 6 has been simplified in order to make it easier to understand. The variables concerning dates, locations and the personages are indicated simply by the letters "d", "l", "p" with their indices, whilst in reality a more complex system of variables and restrictions is needed to account for their actual representation inside the system. In the same way,we have given to the variables *m* and *n* only the values that are most likely in the historical context of RESEDA, without any attempt at generalization. Thus the "jobs" that *p1* may have lost to p2's advantage are limited to monarchical or seigniorial posts directly provided ("SOURCE *n*") by the corresponding authorities. Note that the symbol "v" means "exclusive or", and the symbol "^" means "and" (see the "condition").

The meaning, in clear, of the formalism in figure 7 is as follows (see also figure 4d) : to explain what brought the administration *n* to deprive *p1* of his job - *p1* no longer (end) disposes (BE-AFFECTED-BY) professionally (soc) on the initiative (SOURCE) of *n* of job *m* - the hypothesis suggests we check in the system's memory for the following three facts :

A) at a date contemporary with, or later than, the dismissal of *p1*, the administration *n* gave the post to a second personage p2 (p2 begins to dispose of this post) ;

B) at a date that coincides with, or is previous to, *d1*, the administration *n* comes under the leadership of a personage p3 (*n* starts to have p3 for chief (lid = leader)) ;

C) p2 is p3's "protégé".

Note that, in order to be satisfied, the condition needs the concomitant matching (link "A") of the three search models A, B and C ; the pattern C corresponds to a "relational plane", which is a particular type of plane, without space or temporal information, reserved in RESEDA for the expression of kinship, social or interdependancy relationships (of the kind "protector/protégé" used in C), etc. Of course, in the actual application of the formalism in figure 6, C could be "transformed" into a whole series of search models which will try to match in the base punctual evidence of the dependancy relationship - for example the fact that during some period p2 was an employee of p3, see again Zarri (1981). The hypothesis in figure 6 thus provides a reasonable explanation

Hypothesis concerning the dismissal from a post because of a change in political power

premise : α

α) end+soc+BE-AFFECTED-BY
SUBJ *p1*
OBJ *m : l1*
SOURCE *n : l2*
date1 : *d1*
date2 :

restrictions on the variables of the premise schema :

m = <monarchical-post> v <seigniorial-post>
n = king's-council v seigniorial-council

condition : A ^ B ^ C

A) begin+soc+BE-AFFECTED-BY
SUBJ p2
OBJ *m : l1*
SOURCE *n : l2*
bound1 : i1
bound2 : i2

B) begin+lid+BE-AFFECTED-BY
SUBJ *n : l3*
OBJ p3
bound1 : i3
bound2 : i4

C) (COORD1 p3 p2 (SPECIF protégé))

restrictions on the variables on the condition schemata :

i1 = *d1* < i2
i3 < *d1* = i4

figure 6

of the dismissal, in September 1413, of Philibert de St Léger from his post of baillif of Mâcon, to the advantage of Robert de Bonnay, chamberlain of Charles d'Orléans who took power and the leadership of the royal council in August 1413.

We shall also take into account the two patterns B' and C' of figure 7, which express

the fact that p4, of whom *p1* was the protégé, lost the leadership of the administration *n*. These two patterns are part of a parallel hypothesis having to do with the same general context of the dismissal from a post because of a change in power.

```
B') end+lid+BE-AFFECTED-BY  SUBJ  n : l2
                            OBJ   p4
                            bound1 : 15
                            bound2 : 16
C') (COORD1 p4 p1 (SPECIF protégé))
```

figure 7

If we try to use the patterns of figure 6 and 7 according to the logic of inferences "by transformation", we obtain a result that at first sight is surprising : the six patterns α, A, B, C, B', C' can be equally distributed in four "one-way" transformations. More accurately, a first transformation, without conditions, allows us to go from model α to model A : proof of the fact that *p1* lost post *n* at a date *d1* can be obtained by ensuring that this post was occupied by p2 at the same date *d1* or a later date. A second transformation without conditions which, as the previous one, makes use of the mechanism of the "end/begin" opposition, enables us to transform B' into B. The two remaining transformations, on the other hand, have conditions. One of them enables us to go from search model α to model B', on condition that we can check for the existence of a realisation of model C' : if the search for a direct statement of the dismissal of *p1* by the administration *n* fails, the information provided by the fact that p4, protector of *p1*, left the leadership of *n*, can bring an indirect element to confirm - not decisively - this dismissal. Finally, the parallel transformation with conditions changes search model A into model B, as long as C is verified.

Let us now go back to the hypothesis in figure 4c, that we have already examined in the preceding section. Without going into formal details, it is obvious that, in this case too, one can look at the relationships between premiss and condition as one of the type "transformation" rather than of type "hypothesis". To stay with the example that we have already commented, the information in plane 1 (which corresponds to a particular actualization of the premiss), "Montreuil was opposed to the Burgundian party on the subject of the Hundred Years War", is equivalent to the information in plane 2, "Montreuil is in the Armagnac party" on condition that we know that "Armagnacs and Burgundians were in completely opposite position during the period in question" (plane 3). Planes 2 and 3 provide the result of the use of the search models deduced from the condition in the particular case that we are taking into account.

There is no need here to insist further on this point - other examples of the "hypothesis-transformation" equivalence can be found in Zarri (1981) ; I shall simply point out that, if the semantic content of the hypothesis in figure 4a fully justifies the possibility of seeing it in a "transformation" context, some doubt may appear about the hypothesis in figure 4b. The "semantic distance" between the information which could be retrieved from the premiss-modul (the statement of the gift) and the information brought by the matching of the different search models which actualize the condition in practice, the giver is a parishioner of the religious community, the giver or a relation of the giver has had some important duty inside that community, etc., see Zarri (1979), seems too great to really be able to speak of "automatic equivalence" between the two classes of data. The problem is interesting because it is intrinsically linked to that of marking, if possible, a line between "certain" and "uncertain" transformations, a distinction which is implicit in the very concept of transformation but which is very hard to express in formal terms.

Broadly speaking, the former mainly involve a simple reformulation of the original idea by using other terms of the metalanguage, without modifying the original semantic domain. An obvious example of this is provided by the transformation which translate the common sense rule "if someone goes from one place to another, he has certainly left his starting point", and which allows us to change a formulation in terms of "end+BE-PRESENT" into one in terms of "MOVE", see figure 8.

```
end+BE-PRESENT  SUBJ x : k  ⇨  MOVE SUBJ x : k
                               OBJ  x : l

k = <personage>
k, l = <location>
k ≠ l
```

figure 8

Note that in the terms of RESEDA's metalanguage, the movements of a personage are always expressed under the form of a subject *x* which moves itself as an object.

The second, on the other hand, would have a function analogous in the long run to that which is assigned to the hypotheses, although exercised from a different point of view and according to different practical modalities. This would be to suggest the use of new search models which could lead to information with an interesting logical relationship with that originally searched for, that is to draw "intelligent" parallels which were not *a priori* foreseeable. Of course - see the case of the transformations that could be extracted from the "gift" hypothesis - these parallels are sometimes unexpected and - due to the "inductive" nature of RESEDA's inferences - always conjectural. In the light of these last remarks, the possibility of using the same common sense rules in totally different operational contexts must surely seem less surprising.

CONCLUSION

In the preceding section, I was only concerned with taking a statement originally formulated in a "hypothesis" context in order to adapt it to a "transformation" context. The opposite operation is of course possible, but less interesting in practice mainly for the following two reasons :

- The information content of the common sense rules underlying many transformations, especially "certain" ones, is very small and appears useless when searching for implicit causes, that is when searching for new logical relationships between events that appear *a priori* to be completely independant.

- Moreover, the fact that the new links that RESEDA can find only belong, for the time being, to the category of "causes" - even if this concept is sufficiently general for us : for a description of the "taxonomy of causality" in RESEDA see for example Zarri *et al.* (1980 : 7-11) - impedes the use of a whole series of transformations in the "hypothesis" mode. To quote just one example, the transformation - see Zarri *et al.* (1979 : 49-51) - based on the common sense rule "the redoing, by the mandator or on his order, of some work given to a trustee is proof of an unfavourable attitude of the mandator towards the trustee about the work in question" could be interpreted as a hypothesis, that is to create new semantic links between the data, only in the case of a relationship of the type "at the time of" - which is not explicitly formalized in the current state of the system - between the tangible demonstration of an unfavourable attitude and the act of redoing the work. An interpretation of the type "cause" would really be too arbitrary in this case.

Despite these last remarks, the point that I have tried to demonstrate in this paper, that of the equivalence, "deep rooted" and within certain limits, of the two categories of inference rules used in RESEDA, seems to be sufficiently well based. Beyond the immediate use of such a notion to reach a better formal definition of the system's inference rules - this topic was particularly emphasized in Zarri (1981) - this equivalence allows us to reuse in a completely different context of operation the same "common sense rules" obtained thanks to a patient study of the historian's work. When one thinks of the difficulty involved in establishing an intense, daily collaboration, between two classes of researchers, historians and computer scientists, whose scientific background and methods of work are extremely different, one can appreciate more fairly this precious reuse capability.

REFERENCES

CARBONELL, J.G., Jr. (1978) "POLITICS : Automatic Ideological Reasoning", *Cognitive Science*, II, 27-51.

CHARNIAK, E. (1981) "The Case-Slot Identity Theory", *Cognitive Science*, V, 285-292.

HAFNER, Carole D. (1981) *An Information Retrieval System Based on a Computer Model of Legal Knowledge*. Ann Arbor: UMI Research Press.

KOLODNER, Janet L. (1980) *Retrieval and Organizational Strategies in Conceptual Memory : A Computer Model* (Research Report n° 187). New Haven: Yale University Computer Science Department.

KOLODNER, Janet L. (1981) "Organization and Retrieval in a Conceptual Memory for Events or CON54, Where Are You ?", in *Proceedings of the Seventh International Joint Conference on Artificial Intelligence - IJCAI/81* (Vancouver 1981). Menlo Park (Calif.): The American Association for Artificial Intelligence.

McCARTY, L.T., and SRIDHARAN, N.S. (1980) *The Representation of Conceptual Structures in TAXMAN II, Part One : Logical Templates* (Report LRP-TR-4). New Brunswick (N.J.): Rutgers University Laboratory for Computer Science.

McGREGOR, D.R., and MALONE, J.R. (1981) "The Fact Database : A System Based on Inferential Methods", in *Information Retrieval Research*, Oddy, R.N., *et al.*, eds. London: Butterworths.

ROSNER, M., and SOMERS, H.L. (1980) *Case in Linguistics and Cognitive Science* (Working Paper n° 40). Genève: Fondazione Dalle Molle.

ZARRI, G.P. (1979) "What can artificial intelligence offer to computational Linguistics ? The experience of the RESEDA project", in *Advances in Computer-Aided Literary and Linguistic Research*, Ager, D.E., *et al.*, eds. Birmingham: The University of Aston.

ZARRI, G.P. (1981) "Building the Inference Component of an Historical Information Retrieval System", in *Proceedings of the Seventh International Joint Conference on Artificial Intelligence - IJCAI/81* (Vancouver 1981). Menlo Park (Calif.): The American Association for Artificial Intelligence.

ZARRI, G.P., ORNATO, Monique, LEE, G., LELOUCHE, R., MEISSONNIER, V., and INSOLE, Angela (1979). *Projet RESEDA/1 : Rapport sur les recherches effectuées du 1er mai 1979 au 1er octobre 1979* (Rapp. LISH/150). Paris: Laboratoire d'Informatique pour les Sciences de l'Homme.

ZARRI, G.P., ORNATO, Monique, LEE, G., LELOUCHE, R., MEISSONNIER, V., DE VRIES, P., and ROCCATI, M. (1980). *Projet RESEDA/1 : Rapport sur les recherches effectuées du 1er octobre 1979 au 1er juillet 1980* (Rapp. LISH/187). Paris: Laboratoire d'Informatique pour les Sciences de l'Homme.

AN APPROACH TO THE SYNTAX AND SEMANTICS OF AFFIXES IN 'CONVENTIONALIZED' PHRASE STRUCTURE GRAMMAR

Lenhart K. Schubert

Department of Computing Science
University of Alberta, Edmonton, Alberta T6G 2H1

Abstract

The paper begins with an outline of a concise notation for affixing rules, oriented towards use by a morphological analyzer. After a brief discussion of word constituent parsing based on the output of a morphological analyzer, the paper turns to the problem of semantic analysis of affixes. The problem is considered within the framework of Gazdar's theory of generalized phrase structure grammar for English. This framework is extremely attractive for AI purposes, since it combines a simple, eminently parsable syntax with an equally simple technique (based on the work of Richard Montague) for deriving the logical form of input sentences as a byproduct of the parsing process. The present approach departs from Gazdar's only in the details of the semantic rules. These are formulated so as to yield more or less 'conventional' logical translations of English sentences, rather than higher-order translations in a Montague-style intensional logic. Negative adjective prefixes, noun pluralization, and tense/aspect inflections are looked at in some detail. In each case, the affix logic is found to be inseparable from the logic of larger syntactic structures in which the inflected word is embedded (APs, NPs, and VPs respectively); i.e., logically the affixes do not operate directly on word stems, but on larger structures. In all three cases, however, the goal of obtaining 'conventional' translations seems attainable. For example, semantic rules are sketched for tense/aspect structure which generate predications over time variables, rather than tense logic formulae.

1. Introduction

A very promising recent development in linguistics has been the formulation of context-free grammars for natural languages. The development is surprising, since it runs counter to the transformational school of theoretical linguistics, which has held sway for two and a half decades. Of particular interest from an AI point of view is the work of Gazdar and others on Phrase Structure Grammar (PSG) (Gazdar 1981a, b, Gazdar et al., to appear). PSG does away with transformations, without loss of descriptive economy, by relying on certain metalinguistic devices. Specifically, it represents classes of context-free phrase structure rules by means of rule schemata and metarules, where the latter take phrase structure rules as input and generate new phrase structure rules as output. The details need not concern us here. The essential point is that the object level rules are context-free, allowing the application of efficient context-free parsing algorithms (Thompson 1981, Schubert & Pelletier, to appear). Another crucial advantage lies in the fact that the phrase structure rules are semantically motivated, and are paired with semantic rules that generate the logical translations of the sentences they analyze. Although the translations are not pragmatically disambiguated, they appear to provide a very appropriate point of departure for the pragmatic phase of sentence comprehension.

A phrase structure grammar consists of rules such as the following.

```
<1, [(S DECL) (NP) (VP)], [NP' VP']>
<2, [(NP) (PN)], PN'>,
          PN(2) = {John, Mary, Rome, ...}
<3, [(VP) (V)], V'>,
          V(3) = {run, smile, vanish, ...}
<4, [(VP) (V) (S THAT)], [V' S']>
          V(4) = {believe, notice, remember, ...}
<5, [(S THAT) (that) (S DECL)], S'>
```

Each rule is headed by a rule number, and is followed by a phrase structure rule and the corresponding semantic rule. The phrase structure rules are of more or less conventional type, apart from being given a node-admissibility rather than generative interpretation. For example, the phrase structure part of rule 1 states that an S (sentence) node with feature DECL and direct descendants of category NP and VP is admissible. (In general categories are feature bundles; details have been suppressed here.) Each semantic rule specifies how to form the logical translation of the superordinate node from the logical translations of the subordinate nodes. For example, the semantic part of rule 1 states that the S-translation is to be formed by applying the VP-translation to the NP-translation; the semantic part of rule 4 states that the VP-translation is to be formed by applying the V-translation to the S-translation. (Solid square brackets are used for sentences in infix form and broken brackets for non-sentential predicate expressions in prefix form. These conventions are purely cosmetic -- they yield very readable translations.) In addition, for each rule introducing a lexical category there is a specification of the subcategory of lexemes allowed in the rule. The sample rules can account for such sentences as "John notices that Mary smiles". The structure assigned to this sentence by the rules can be indicated by bracketing and rule numbers as follows:

```
[1[2 John] [4 notices [5 that
                [1[2 Mary] [3 smiles]]]]]
```

The target logic for the semantic rules is standardly Montague's intensional logic (Montague 1970a, b, c). For AI purposes, however, it would be preferable to have a more conventional target logic, such as a second order modal predicate logic. The reasons are that conventional logics have a more natural semantics (e.g., in Kripke semantics names denote individuals rather than property sets), make inference more tractable, and require fewer meaning postulates. The expressive adequacy of conventional logics was called into question by Montague, particularly with regard to intensional locutions such as "John looks for a unicorn", "John conceives of a unicorn", and "John worships a unicorn". However, Schubert & Pelletier (to appear) argue that such locutions admit conventional translations involving modal operators, and show how to reformulate Gazdar's semantic rules to yield such translations.

The purpose of the present paper is to extend this conventionalized PSG framework to deal with some aspects of affix structure and meaning. Such extensions are important not only because they unburden the lexicon used by a parser, no longer requiring it to list inflectional and derivational variants of every lexeme, but also because they provide a basis for guessing categories and meanings of unknown words. The syntactic formalism proposed herein for affix structure is easily "compiled" into a form suitable for efficient affix analysis. Logical translation rules are offered for a few affix types, namely negative prefixes, noun plurals, and tense/aspect inflections.

2. Affixation and word constituent structure

The form of affix rules is illustrated by the following rule set, which covers regular plurals as well as the more important irregular plurals:

rule	examples
⟨let⟩sh,s,x,y,z⟩(/s)	aces, strifes, skis, radios, cargos, oafs, beliefs, reefs, proofs, cliffs, gulfs, scarfs
⟨vow⟩⟨a|e|o|l|r⟩(f/ves)	leaves, loaves, thieves, hooves, calves, elves, wolves, scarves
⟨c|s⟩h(/es) ⟨let⟩c,s⟩h(/s)	churches, bushes blahs, boughs, sylphs, baths
(ife/ves)	knives, wives, lives
⟨cons⟩o(/es)	heroes, cargoes
⟨s|x⟩(/es)	lenses, kisses, buses, foxes
⟨vow⟩⟨s|z⟩(/*es)	busses, quizzes
⟨cons⟩z(/es)	blitzes, fizzes
⟨cons⟩y(/ies)	flies
⟨vow⟩u⟩y(/s)	days, keys, boys
⟨let⟩q⟩uy(/s)	guys
quy/ies)	soliloquies
⟨-⟩⟨char⟩('s)	i's, 5's

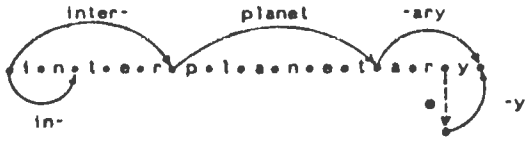
Each rule consists of a sequence of simple or compound character predicates and substitution rules. For example, the first rule specifies that if a word (noun stem) ends in a letter which is neither an h, s, x, y, or z, then the empty character at the end of the word is to be replaced by an s to form the plural. The second rule applies to a word whose third last letter is a vowel, whose second last letter is a, e, o, l, or r, and whose last letter is f; the plural is formed by replacing the f by ves. The remaining rules are more or less self-explanatory. (In the last rule ⟨-⟩ is a predicate signifying a morpheme boundary.) Further rules (even word-specific rules) can easily be added. For example, the null plural suffix for "fish", the irregular plural for "mouse" and "louse", and the Latin masculine plural can be expressed as

⟨-⟩fish	fish, whitefish
⟨-⟩⟨l|m⟩(ous/ice	lice, fieldmice
⟨vow⟩⟨cons|⟩⟨cons|⟩⟨cons|⟩(us/i)	dei, foci, cacti

Here ⟨cons|⟩ signifies a consonant or an empty character.

Such rules can be converted automatically into a decision tree suitable for morphological analysis. A PASCAL program has been written to do this in a way which permits easy editing of the rule set. The root of the tree corresponds to the end of the word to be analyzed and the branches correspond to successive tests applied to successive characters of the word in a right-to-left scan. The tests (both simple and compound) are efficiently implemented as comparisons against bit vectors. The resultant flow of control during suffix analysis is much the same as in a directly programmed 'morpher' such as that of Cercone (1977). Prefixes can be dealt with in much the same way.

The type of morpher output that will be assumed is similar to that discussed by Kay (1977) and is illustrated by the following analysis of "interplanetary":



The node reached by the subsidiary character chain labelled "e" would become an arc destination if the lexicon contained (or the context leads to postulation of) a verb or noun "planetare" transformable to "planetary" (cf., scare –> scary).

In addition, the morpher (aided by the lexicon) supplies the categories of the word constituents, such as

(inter- A/A N/N V/V), (in- A/A V/V), (planet N), (-ary A/N N/N ...), (-y A/V A/N N/N).

Here N/N is the affix category which forms nouns from nouns, A/V the category which forms (deverbal) adjectives from verbs, etc. Dots indicate potentially incomplete categorical knowledge.

From the morpher output, the parser is assumed to produce phrase structure 'trees' such as

[A [A/A inter-] [A [N planet] [A/N -ary]]]

using phrase structure rules such as [A N A/N] and [A A/A A].

The scene is now set for defining logical translation rules for affixes. This task is far more challenging than defining affix syntax, since affixes vary so widely in semantic function. Therefore, this paper focuses on the three particular (but important) cases of negative adjectival prefixes, noun plurals, and tense and aspect inflections.

3. 'Conventional' semantic rules

The translation of a word is to be constructed much as the translation of a sentence is constructed in PSG, i.e., by combining operators with operands in an order determined by the parse tree. Above, the translation of "-ary" would be applied to the translation of "planet", and then the translation of "inter-" would be applied to the result to yield the expression translating "interplanetary".

At this point the rules for constructing logical translations in accordance with specific semantic rules need to be recapitulated.

The logical translations of individual lexical items (usually single words) are individual constants, predicators, functors, and quantifiers; e.g., the translations of "Mary", "boy", and "loves" might be the constant Mary2, the monadic predicator boy4, and the dyadic predicator loves1 respectively. The numerical indices 1, 2, 3, ... in such translations are not obtained from the lexicon, but rather are affixed after lexical retrieval. Strictly, the resulting indexed symbols are not to be regarded as proper logical symbols, but rather as preliminary translations which may be ambiguous. They are to be replaced in a later postprocessing phase by unambiguous logical symbols such as MARY17, BOY1, and LOVES. A crucial constraint imposed by the preliminary translations is that identical symbols (and in general, identical expressions) must be identically disambiguated. for example,

[John2 shaves1 John2]

can only mean that John shaves himself, whereas

[John3 shaves1 John2]

can mean that one individual named John shaves another individual named John. Analogous remarks apply to

[<some2 man3> shaves1 <some2 man3>]
versus
[<some4 man5> shaves1 <some2 man3>].

In the semantic rules, expressions of form [E2 E1], ⌈E1 E2⌉, (E1 E2), and ⟨E1 E2⟩ all denote combination of an expression E1, as operator, with an expression E2, as operand. The resultant expression in the first case is a sentence (in infix form), in the second a predicate expression (in prefix form), in the third a functional expression (in prefix form), and in the fourth a quantifier expression (consisting of a quantifier followed by one or more predicate expressions constraining the quantifier).

Both in the semantic rules and in the translations they produce, application of an operator to a k-tuple of operands is regarded as equivalent to applying the operator to the first operand, then applying the resulting expression to the second operand, and so on.[1]

Examples:

1. With E1 = gives1, E2 = Mary2, E3 = ⟨a3 dog4⟩,
   ⌈E1 E2 E3⌉ = ⌈[E1 E2] E3⌉
   = ⌈[gives1 Mary2] ⟨a3 dog4⟩⌉
   = ⌈gives1 Mary2 ⟨a3 dog4⟩⌉.

2. With E1 = ⌈gives1 Mary2 ⟨a3 dog4⟩⌉, E2 = John5,
   [E2 E1] = [John5 ⌈gives1 Mary2 ⟨a3 dog4⟩⌉]
   = [John5 gives1 Mary2 ⟨a3 dog4⟩].

3. With E1 = ⟨the2 (big3 man4)⟩, E2 = ⌈from5 NY6⌉,
   ⟨E1 E2⟩ = ⟨⟨the2 (big3 man4)⟩ ⌈from5 NY6⌉⟩
   = ⟨the2 (big3 man4) ⌈from5 NY6⌉⟩.

In addition to the above types of operation, the semantic rules can also specify lambda abstraction. In particular, an expression of form λxE specifies that a new variable is to be substituted for all occurrences of x in E and the resultant expression prefixed with λ followed by the new variable. Also, in the above rules for operand application, immediate lambda conversion is to be carried out if the operator is a lambda abstract.
Example:
With E1 = [John5 gives1 x ⟨a3 dog4⟩], E2 = Mary6,
⌈λxE1 E2⌉ = ⌈λy[John5 gives1 y ⟨a3 dog4⟩] Mary6⌉
= [John5 gives1 Mary6 ⟨a3 dog4⟩].

In the pragmatic post-processing phase which follows translation, quantifier expressions in term positions are replaced by variables and moved to the head of a sentence in which they were formerly embedded; at this stage quantifier scope ambiguities are resolved. For example,

[<some4 boy5> loves1 <every2 girl3>]

becomes either

(some4 x: boy5) (every2 y: girl3) [x loves1 y]
or
(every2 x: girl3) (some4 y: boy5) [y loves1 x].

Note that predicate expressions such as boy5 and ⌈loves1 Mary2⌉ are equivalent to λx[x boy5] and λx[x loves1 Mary2] respectively, assuming that boy5 and loves1 are one- and two-argument predicates respectively.

[1] This equivalence assumption can be justified semantically by interpreting operator applications in terms of sections of relations.

4. Negative adjective prefixes

The present task is to formulate semantic rules of the above type for affixes, beginning with the relatively simple case of negative adjective prefixes. These are
a-, an-, dis-, il-, im-, in-, ir-, non-, and un-.

What is of interest is the systematic component of the meaning conveyed by these prefixes. The specialized meanings of lexemes such as "disgraceful" (cf., "ungraceful"), "impertinent", "unflinching", etc., is a side issue. Also, it is important not to confuse adjective prefixes with verb prefixes. For example, dis- is an adjecte prefix in "disagreeable" and "discontented" but a verb prefix in "discouraged" and (probably) "displeased".

The first task is to determine the logical category of the operands of the prefixes in question. The main possibilities are illustrated by the following examples:
(a) John is unconscious;
John is illiterate;
(b) John is dissatisfied with Mary's work;
John is unwilling to leave;
(c) John is an uneducated boor;
John swallowed some nontoxic paint;
(d) John bought a non-genuine antique;
John is an atypical student;
John is an unskillful surgeon.

In the (a) examples, there is little doubt that the negative prefixes are predicate modifiers. An appropriate syntactic-semantic prefixing rule is

⟨6, [(AP PRED) (A/A F) (AP PRED F)], (A/A' AP')⟩,
A/A(6) = (F), F ∈ {a-, an-, ..., un-}.

Here the prefix itself is used as an agreement feature to ensure that the correct choice of prefix will be made for every admissible adjective. But observe that the rule combines the prefix not with the adjective, but with an AP (adjective phrase); the latter is formed from the adjective by the rule

⟨7, [(AP PRED F) (A PRED F)], A'⟩.

The reason for this approach will be stated shortly. The feature PRED picks out adjectives that can appear in predicative position. In these cases, then, the prefix acts semantically as a function from predicate meanings to predicate meanings.

In the (b) examples, it is unclear whether the operands are predicates such as "satisfied with Mary's work" and "willing to leave" or just the predicate-forming operators "satisfied" and "willing". The trouble with the latter analysis is that it treats those prefixes which are applicable to a variety of adjectives with disparate complement requirements -- viz., most or all of them -- as multivocal; for it is hard to see how one and the same semantic function could have both predicates and various predicate-forming operators as arguments. While some categories of English words may be genuinely multivocal (see the discussion of numerals below), the linguistic evidence in the present case is against multivocality. For example, if un- were multivocal, a sentence like
John is unenthusiastic about the job
should be ambiguous between a reading according to which John does not feel enthusiastic about the job and another according to which his feeling about the job is one of 'non-enthusiasm'; such is not the case.

Therefore the first analysis has been adopted, according to which the negative prefixes operate on complete (monadic) predicates. It is to accommodate this analysis that the prefix operand in rule 6 was chosen to be an AP. If the analysis is correct, it implies that adjectives with negative prefixes (even directly lexicalized ones) do not deliver their meaning all at once; the meaning of the stem is deployed first, allowing it to combine with complements; meanwhile, the negative force of the prefix remains encapsulated in the prefix feature, to be released only when an AP has been formed. Similar delayed action effects are implicit in some of Gazdar's rules, such as those for coordination.

In the (c) examples the prefixes appear to operate on predicate modifiers. However, the result of the operation is intuitively a conjunction: an "uneducated boor" is someone who is uneducated and a boor, and similarly "nontoxic paint" is stuff that is nontoxic and paint. Thus the prefixes actually operate on predicates, and no new prefix rule is needed. The rule which translates certain adjectives in attributive position as predicates conjoined with the noun is roughly the following (restated more accurately as rule 10 below):

⟨8, [[AN] [AP PRED] [N]], λx[[x AP'] & [x N']]⟩.

In the (d) examples, however, the negated adjectives are genuine predicate modifiers: a "genuine antique" is not an article that is genuine and an antique, a "typical student" is not someone who is typical and a student, and a "skillful surgeon" is not someone who is skillful and a surgeon. Nevertheless a reduction to the predicative case, similar to that for the (b) cases, seems possible: in "non-genuine antique", "atypical student" and "unskillful surgeon", the prefixes can be viewed as operating logically on the predicates "genuine antique", "typical student" and "skillfull at surgery" producing results similar to "not a genuine antique", "not a typical student" and "not skillful at surgery" respectively. Note, however, that (in contrast with the case of a non-genuine antique), an atypical student is still a student and an unskillful surgeon is still a surgeon; furthermore, in cases like that of the unskillful surgeon, it is possible to imagine contexts (e.g., a medical school soccer play-off) in which the skills referred to are not those determined by the noun on which the adjective operates. Syntactic-semantic rules can be formulated which correctly make these distinctions, rendering the second and third (d)-examples as conjunctions similar to "John is not a typical student and is a student" and "John is not skillful at something (pragmatically, at surgery) and is a surgeon". The important point for present purposes is that the negative prefixes are treated uniformly as predicate modifiers.

Of course, a full semantic investigation of negative prefixes requires more than an analysis of their role in the mapping from surface syntax to logical form; it also requires formulation of axioms capturing their content. A key question concerning the negative prefixes is to what extent their semantic import differs from that of negation. All of them entail negation (an apolitical person is not a political person, a disagreeable odour is not an agreeable odour, an incomplete success is not a complete success, and so on) but is the converse generally true as well? A perusal of dictionary entries reveals few convincing counterexamples. The following are probably as good as any:

- an odour that is not agreeable need not be disagreeable; it may simply be neutral;
- conduct that is not moral need not be immoral but may simply be amoral (!)
- a person who is not kind need not be unkind but may merely be aloof.

However, since such cases are relatively scarce, they can be treated as exceptions whose exact meaning cannot be reconstructed from the meanings of the prefix and stem. If their meanings were systematically determined, the following rather similar examples should make equally good sense:

- a person who is not honest need not be dishonest;
- two situations that are not similar need not be dissimilar;
- drinking that is not moderate need not be immoderate;
- a remedy that is not effective need not be ineffective;
- one who is not afraid need not be unafraid;
- conduct that is not fair need not be unfair.

But they are not; a certain mental effort is required to make any sense of them, indicating that some adjustment of the usual meanings is required. Perhaps these examples induce a bifurcation of meanings, or separation of meaning components, much as the following examples do:

John is the loser, yet he is not.
John hates Mary, yet he doesn't.
Being helped by John is not being helped, but hindered.
There are wines, and then there are wines.

Also, the situation is very likely complicated by meaning overtones such as implicatures and connotations. It may well be, for example, that application of the dis- operator to a degree adjective implicates a more drastic inversion of attributes than mere negation. However, the focus here is on bare logical content, i.e., on the content relevant to truth conditions; and on Grice's analysis, truth conditions are independent of implicatures (Grice 1975).

The upshot is that to a first approximation, all of the negative adjectival prefixes can be translated in terms of a common negative operator, say "non", related to sentential negation by the axiom schema

≡[(non P) ≡ λx¬[x P]].

Thus, the property of being uneducated is the property of being an individual that is not educated, the property of being dishonest is the property of being an individual that is not honest, etc. The given schema is entirely compatible with the manifest non-synonymy of such phrases as "completely uneducated" and "not completely educated", since (completely λx¬[x educated]) is not equivalent to λx¬[x (completely educated)].

In a similar way, many other affixes (especially derivational affixes) can be treated logically as predicate modifiers.

5. Noun plurals

The logic of negative prefixes was seen to be closely bound up with the logic of larger constructions, especially of APs and NPs, and for this reason non-transparent. The same is true of noun plurals. In the first place, it is to be expected that VP pluralization will interact logically with noun pluralization in plural NP+VP sentences. Moreover, pluralization does not correspond simply to application of some logical operator to the translation of the pluralized noun, as the following sentence demonstrates:

The serrate leaves of the old oak covered the ground.

Clearly the sentence makes reference to a plurality each of whose members is a serrate leaf, rather than to a serrate plurality of leaves; i.e., the plural applies logically to "serrate leaf", not to "leaf". Thus the effect of pluralization, like that of negative prefixes, is in general delayed.

The above sentence also illustrates another point: plurals can give rise to alternative readings, corresponding to distributive and collective interpretations of the NP. Though the only natural reading of the sentence happens to be the collective one, it can in principle be taken to assert that each leaf individually covers the ground.

As far as the semantics of pluralities (collections, ensembles, ...) is concerned, a theory like that of Link (1982) seems appropriate; i.e., pluralities are individuals which are 'sums' of other individuals. The 'sum' of two pluralities equals the 'sum' of their individual constituents, and a 'sum' of one atomic constituent equals that atomic constituent (cf., Bunt 1979 and Moore 1981). Pluralities of atoms are to be distinguished from their 'material fusion'. Link uses an operator "*" to convert predicates over atoms to predicates over corresponding pluralities, and ß to form predicates over proper pluralities of two or more atoms. As an aid to intuition ß will be written "two-or-more" here. For example, (two-or-more leaf) is a predicate over pluralities of leaves; or putting it a little more explicitly, it is a predicate that applies truly to objects with two or more atomic constituents each of which is a leaf. The English

numerals one, two, three, ... can be interpreted analogously as predicate operators which generate predicates over pluralities; thus (two leaf) is a predicate truly applicable to any plurality of (exactly) two leaves.

Both the collective-distributive ambiguity, and an implicit quantifier ambiguity arises in the following sentence:

Two people can paddle a canoe.

The following paraphrases resolve the implicit quantifier ambiguity (while preserving the collective-distributive ambiguity):

Any two people can paddle a canoe.
There are two people who can paddle a canoe.

(There is an analogous ambiguity about the phrase "a canoe", which could mean "any canoe" or "some canoe", but this is not of concern here.) Now, the first ("any") reading clearly calls for quantification over pluralities of two people, in the manner

(∀x: (two person))...

It then also seems natural to render the second reading as

(∃x: (two person))...

Now, this translation seems well suited to expressing the collective version of the existential reading; but how is the distributive reading to be expressed, to the effect that there are two people each of whom can paddle a canoe? Link would employ the pluralized predicate (* can paddle a canoe) here, obtaining the equivalent of

There is a plurality of two people,
with the property that each atomic
constituent can paddle a canoe.

But here a subtle dilemma arises. In view of the close syntactic similarity of phrases like

a person, some person,
one person, two persons, ...

it would also seem natural to translate the numerals directly as quantifiers, in this manner:

(2x: person)...,

i.e., "At least two persons are such that each of them ...". Moreover, this would give a substantially simpler translation of the distributive existential reading of the sentence under consideration than the sort of translation indicated above, involving the two pluralizing operators "two" and "*". Should numerals therefore be treated as bivocal? Experimentation with syntactic-semantic rules of NP formation has failed to turn up convincing evidence for or against bivocality. However, the following observation has helped to tip the scales in favour of a bivocal treatment of numerals. If Link's treatment were correct, the "canoe" sentence should be four ways ambiguous:

Any		each	
	two people can		paddle a canoe.
Some		together	

But there is no "any"-"each" reading. Now, the "any" reading of the NP is indispensable, hence it is the "each" reading of the VP which must be rejected. In other words, the VP is unambiguous, and involves no pluralizing operator. Of course, this analysis also gets rid of the "some"-"each" reading, but this is precisely the reading re-instated by translating "two" as a quantifier.

There is a close connection between the presumed bivocality of the numerals (and certain related adjectives including "many", "few", "countless" and "numerous") and the behaviour of "and" in such sentences as

John and Mary can paddle a canoe.

Again, the choice is between treating the NP as univocal and the VP as ambiguous between "each" and "together" readings, or treating the NP as ambiguous and the VP as univocal. Only the latter approach is consistent with the uniform treatment of and/or-coordination proposed elsewhere as part of a 'conventionalized' version of Gazdar's grammar (Schubert & Pelletier, to appear).

The following are some NP translations in keeping with the preceding discussion. They will be followed by some remarks on the NP grammar needed to generate them. P abbreviates λv[[v juicy] & [v apple]], Q denotes the appropriate VP translation in the first three examples, and ‹ is the predicate which relates atoms to pluralities containing them. The NP translations are shown as they would appear in the parser output (minus indices)', i.e., prior to variable insertion and quantifier extraction.

five juicy apples had worms in them.
  [‹5 P› Q]
five juicy apples cost a dollar nowadays.
  [‹∀ (five P)› Q]
five juicy apples filled the basket.
  [‹∃ (five P)› Q]
the five juicy apples
  ‹the (five P)›,
  ‹∀ ¦‹ ‹the (five P)›¦›
the juicy apples
  ‹the (two-or-more P)›,
  ‹∀ ¦‹ ‹the (two-or-more P)›¦›
five of the juicy apples
  ‹5 ¦‹ ‹the (two-or-more P)›¦›,
  ‹∀ (five ¦‹ ‹the (two-or-more P)›¦)›,
  ‹∃ (five ¦‹ ‹the (two-or-more P)›¦)›
all (of) the juicy apples
  ‹∀ ¦‹ ‹the (five P)›¦›,
  ‹the P›
all (of the) five juicy apples
  ‹∀ ¦‹ ‹the (five P)›¦›,
  ‹the (five P)›
all five of the juicy apples
  ‹∀ ¦‹ ‹the (five ¦‹ ‹the (two-or-more P)›¦)›¦›,
  ‹the (five ¦‹ ‹the (two-or-more P)›¦)›.

After variable insertion and quantifier extraction the first and last examples would become
  (5x: P)[x Q] and
  (the z:(five λw(the y:(two-or-more P))[w ‹ y]]), or
  (the y:(two-or-more P))(the z:(five λw[w ‹ y]]),
respectively.

The NP rules needed to produce these translations begin with the plural affix rule

  ‹9, [(N PLUR) (N SING) (N/N PLUR)], N'›,

which merely introduces the PLUR feature while taking the translation of the plural noun to be the same as that of the singular noun. The remaining rules fall naturally into seven groups: those which add premodifiers (as in "wide angle lens"); those which add an AP other than a numeral or ordinal (as in "large, beautifully landscaped garden"); those which add a numeral (as in "five heavy snowfalls"); those which add an ordinal (as in "first five heavy snowfalls"); those which add a determiner (as in "the juiciest apple"); those which add a predeterminer (as in "all the apples"); and those which add postmodifiers (as in "the apple on the table which I left for you ").

In the formation of a NP from a N, intermediate AN ("adjectives plus noun") combinations are formed. The constraints governing the addition of premodifiers, APs, numerals, ordinals and determiners can be formulated in terms of features added to the AN by these constituents. Features that appear to play a central role are PRED (carried by adjectives like "juicy" which are allowed in predicative positions), ATTR (carried by adjectives like "consummate" allowed only in attributive positions), NUM (carried by numeral adjectives like "five" and non-extreme ordinals like "fifth"), ORD (carried both by extreme ordinals like "first", "next", "last", and "only", and by non-extreme ordinals), COMP (carried by comparative adjectives like "more excited"), and SUP (carried by superlative adjectives like "juiciest").

For example, an (AP ORD) can be added to the AN only as long as the features ORD, COMP and SUP are not yet present; (witness *the first juiciest apples, *the first five second apples). An important semantic aspect of the ORD and SUP features is that their introduction into the syntax is accompanied by the introduction of sentence schemas conjoined with the AN into the AN-translation; these schemas contain distinguished predicate variables (P and Q below) which are ultimately bound to the postmodifiers of the NP, if any. Consequently, in a phrase like "the juiciest apple on the table", the scope of the superlative adjective will encompass the postmodifying PP, while in a non-superlative case like "the juicy apple on the table", the scope of the adjective excludes the postmodifier.

A few tentative NP rules are

```
<10, [(AN PRED ¬ATTR ¬NUM ¬ORD ¬SUP)
      (AP PRED ¬SUP) (AN ¬PRED ¬ATTR ¬NUM ¬ORD)],
      λx[[x AP'] & [x AN']]>
<11, [(AN PLUR NUM ¬ORD) (AN PLUR ¬NUM ¬ORD)],
      (two-or-more AN')>
<12, [(AN NUM ¬ORD) (AP NUM) (AN ¬NUM ¬ORD)],
      (AP' AN')>
<13, [(AN PLUR NUM ORD)
      (AP ¬NUM ORD) (AN PLUR NUM ¬ORD ¬COMP ¬SUP)],
      (ORD' λx[[x AN'] & [x P] & [x Q]])>
<14, [(NP NUM) (DET NUM) (AN ¬NUM ¬ORD ¬SUP)],
      <DET' λx[[x AN'] & [x P] & [x Q]]>>
<15, [(NP) (PP)], ['APNP' PP'];>.
```

Rule 10 provides the conjunctive translation of a potentially predicative AP in attributive position. Rule 11 essentially translates a null numeral as "two-or-more" (but without actually postulating a null constituent). Rules 12 and 14 respectively treat a numeral as an adjective and as a determiner, translating it as a predicate modifier in the first case and as a quantifier in the second. Rule 13 introduces an ordinal, treating it as a modifier of predicates over pluralities, and adding conjuncts to the translation which are to be bound by postmodifiers; thus the scope of the ordinal will encompass the postmodifiers. In the same way rule 14 ensures that the scope of the numeric quantifier will encompass the postmodifiers. Rule 15 binds the postmodifying PP to the (originally free) predicate variable P.

6. Tense and aspect

The final topic to be looked at, more superficially than the others, is the semantics of tense/aspect inflections.

Once again, there is a close interaction between the syntax and semantics of the inflections in question and the syntax and semantics of the structures in which the inflected words are embedded. In the present case it is the grammar of the VP, and in particular of the auxiliary system, which provides the context for the inflectional paradigm. The reader is referred to Gazdar, Pullum & Sag (1980) for a comprehensive auxiliary system grammar based on mutual constraints among VP features associated with the auxiliaries. All that is needed for present purposes is the assumption that suffix analysis together with syntactic recognition of "be", "have", "will", and "be going to" auxiliaries is capable of recognizing the following types of syntactic constituents: PRES, PAST, PROG (progressive aspect: "be V-ing"), PERF (perfective aspect: "have V-en"), ABSF (absolute future: "will V"), and RELF (relative future: "be going to V").

Traditionally, tense and aspect have been analyzed logically by associating an event time, one or two reference times, and a time of speech with any declarative sentence, and postulating a particular set of relationships among these times for each possible tense-aspect configuration. This approach is due to Reichenbach (1947) and has frequently served as a basis for analyses of tense in AI (e.g., see Bruce 1972, Kahn 1975, and Cohen 1977). For example, a past perfect construction such as "He had laughed" is said to place the event time before the reference time which is in turn prior to the time of speech. For a case like simple past, where intuitively only two times are being related, the reference time is said to coincide with the event time. For the present tense, all three times are said to coincide.

From the point of view of phrase structure grammar, this approach is quite unnatural; for it fails to show how the meaning of a VP in the temporal dimension is built up incrementally, in parallel with its syntactic construction. For example, how is the meaning of "will have laughed" functionally determined by the meaning of "will" and "have laughed"?

Montague (1970c) incorporated future tense and present perfect into his fragment of English in a way which is indeed compositional. For this purpose he used intensional operators intuitively expressing "it will be the case that" and "it has been the case that". Thus, "John will have laughed" is intuitively expressed as "It will be the case that it has been the case that John laughs". The formal semantics is in keeping with Reichenbach's analysis in this case, but in general there is no fixed set of time points which a sentence relates; instead, one new time reference is introduced by each application of a tense operator. A similar approach is taken in Schwind's tense logic (Schwind 1978). Guenthner (1978) extends Montague's (essentially Priorian) tense logic to deal with time intervals.

The present framework for determining the logical form of English sentences is expressly designed to stay with conventional logic as far as possible. Since tense operators are non-standard, the question arises whether translation rules can be formulated which treat time conventionally, i.e., which introduce time variables and express time relationships as predications over these variables. Such an approach would also have computational advantages; to date most (perhaps all) language understanding systems which have attempted to deal seriously with time have relied on explicit representations of moments and intervals of time to facilitate inference of time relationships.

The following is a sketch of such an approach. Intuitively, any tense or aspect functor is to be viewed as mapping a given input time into an output time, while simultaneously generating a constraint relating those times. The constraints are of the following type, where in each case t is the input time variable and t' or now' the output time variable (and now' evaluates to the time of speech):

```
PRES: [t ends-at now']
PAST: [t before now']
PROG: [t spans t']
PERF: [t before t']
ABSF: [t after now']
RELF: [t after t'].
```

For example, suppose that t0 denotes the time of John's laughing in "John will have laughed". Thus the VP is of form [ABSF [PERF [laugh at t0]]], and so PERF is applied first to t0, producing output time t1 (say) and constraint

[t0 before t1].

Next ABSF is applied to t1, producing output time now2 (= time of speech) and constraint

[t1 after now2].

In effect, t1 is the Reichenbachian reference time in this example, but this time has been generated as a byproduct of the stepwise translation process. Note that no reference time will be generated for simple present, past and future. Proper time relations are automatically generated for constructions such as "Mary had laughed", "Mary had been laughing", "Mary had been going to laugh", "Mary will have been going to be laughing", etc. Note that in the last two examples two and three 'reference times' are produced, and quite properly so. (The constraints specified for the tense/aspect functors above are not always the correct ones, but they are among those most frequently intended. For example, there is a progressive reading of PRES and a future reading of PROG.)

Technically, the semantic rules for the tense/aspect functors can be expressed in the same way as other rules. In effect, they recast a sentence such as "John will have laughed" as "John laughs at some time before some time after now". The variable generation process described in intuitive terms above is accomplished by existential quantification. To illustrate the mechanism, here are the rules for PERF and ABSF (ignoring details of features):

```
<16, [(VP PERF) (V have) (VP -en)],
     [λT'VP' [before <[ T>]]>
<17, [(VP ABSF) (V will) (VP BASE)],
     [λT'VP' [after now']]>.
```

T is a distinguished time-predicate variable. When a temporal VP is first formed, prior to incorporation of the auxiliaries, it is immediately applied to a time argument <[T>. Thus, prior to the application of PERF and ABSF, the translation of "laughed" in the sentence under consideration will be

```
[laughs <[ T>]
```

When the PERF translation rule is applied to this, the result is

```
[λS[laughs <[ S>] [before <[ T>]]
= [laughs <[ [before <[ T>]]]
```

(Note the renaming of T to S in the lambda abstract, in accordance with the rules described earlier.) Similarly, when the ABSF rule is applied, the result is

```
[laughs <[ [before <[ [after now2]>]]>].
```

Ultimately, when explicit variables are introduced and the quantifiers extracted from the sentence matrix, the sentence translation becomes

```
([t1: [after now2]] ([t0: [before t1]]
                     [John laughs t0]);
```

With [after now2] = λx[x after now2], etc., and with quantifier restrictions converted to conjuncts, the result is

```
([t1][[t0][[t1 after now2] & [t0 before t1]&
          [John laughs t0]],
```

which is the sort of conventional translation frequently presupposed in knowledge representation systems (e.g., Schubert, Goebel & Cercone 1979).

7. Conclusion

These extensions further substantiate the thesis that logical form can be computed as a byproduct of parsing (as Montague maintained) and further, that the target logic can be kept more or less conventional. If true, these claims should prove very profitable for AI, since they imply that the natural language understanding process (at least in the initial stages) is simpler and more systematic than has generally been supposed.

Acknowledgements

The author gratefully acknowledges Jeffry Pelletier's helpful comments on several issues and well-informed pointers to the literature. The research was supported by the National Sciences and Engineering Research Council of Canada under grant A8818.

References

Bruce, B. C. (1972). A model for temporal references and its application in a question answering program. Artificial Intelligence 3.

Bunt, H. C. (1979). Ensembles and the formal semantic properties of mass terms. In Pelletier, F. J. (ed.), Mass Terms: Some Philosophical Problems, D. Reidel, Dortrecht, Holland, 249-277.

Cercone, N. (1977). A heuristic morphological analyzer for natural language understanding programs. Proc. of 1st Int. IEEE COMPSAC '77, Chicago, IL, 676-682.

Cohen, R. (1977). Computer analysis of temporal reference. Tech. Rep. No. 107, Dept. of Computer Science, Univ. of Toronto, Toronto, Ont.

Gazdar, G. (1981a). Phrase structure grammar. To appear in Jacobson, P., and Pullum, G. K. (eds.), The Nature of Syntactic Representation, D. Reidel, Dortrecht.

Gazdar, G. (1981b). Unbounded dependencies and and coordinate structure, Linguistic Inquiry 12.2.

Gazdar, G., Pullum, G. K., and Sag, I. (1980). A phrase structure grammar of the English auxiliary system. MS. To appear as "Auxiliaries and related phenomena" in Language.

Gazdar, G., Klein, E., Pullum, G. K., and Sag, I. English Syntax, to appear.

Grice, H. P. (1975). Logic and conversation. In Davidson, D., and Harman, G. (eds.), The Logic of Grammar, Dickenson, Encino, CA, 64-75.

Guenthner, F. (1978). Time schemes, tense logic, and the analysis of English tenses. In Guenthner, F., & Schmidt, S. J. (eds.), Formal Semantics and Pragmatics for Natural Languages, D. Reidel, Dortrecht, 201-222.

Kay, M. (1977). Morphological and syntactic analysis. In Zampolli, A. (ed.), Linguistic Structures Processing, North-Holland, Amsterdam, 131-234.

Kahn, K. M. (1975). Mechanization of temporal knowledge. MIT Project MAC Tech. Rep. 155, MIT, Cambridge, MA.

Knuth, D. E. (1968). Semantics of context-free languages. Mathematical Systems Theory 2, 127-145.

Link, G. (1982). A logical analysis of plurals and mass terms: A lattice-theoretical approach.

Montague, R. (1970a). English as a formal language. In Thomason (1974), 188-221.

Montague, R. (1970b). Universal grammar. In Thomason (1974), 222-246.

Montague, R. (1970c). The proper treatment of quantification in ordinary English. In Thomason (1974), 247-270.

Moore, R. C. (1981). Problems in logical form. 19th Ann. Meet. of the Assoc. for Computational Linguistics, June 29 - July 1, Stanford Univ., Stanford, CA, 117-124.

Schubert, L. K., Goebel, R., & Cercone, N. (1979). The structure and organization of a semantic net for comprehension and inference. In Findler, N. V. (ed.), Associative Networks: The Representation and Use of Knowledge by Machines, Academic Press, New York, NY, 121-175.

Schubert, L. K., and Pelletier, F. J. (to appear). From English to logic: context-free computation of 'conventional' logical translations. MS, Dept. of Computing Science, Univ. of Alberta, Edmonton, Alberta.

Schwind, C. (1978a). A formalism for the description of question answering systems. In Bolc, L. (ed.), Natural Language Communication with Computers, Springer-Verlag, Berlin, Heidelberg & New York, 1-48.

Schwind, C. (1978b). The translation of natural language texts into state logic formulae. Tech. Rep. TUM-INFO-7806, Technische Univ. Muenchen (available from Bibliothek des Fachbereichs Mathematik, Technische Univ. Muenchen, D-8000 Muenchen 2, W. Germany).

Thomason, R. H. (ed.) (1974). Formal Philosophy: Selected Papers of Richard Montague. Yale Univ. Press, New Haven, CT.

Thompson, H. (1981). Chart parsing and rule schemata in PSG. Proc. 19th Ann. Meet. of the Assoc. for Computational Linguistics, June 29 - July 1, Stanford Univ., Stanford, CA, 167-172.

VERBS IN DATABASES*

David Maier
Sharon C. Salveter

Computer Science Department
SUNY Stony Brook
Stony Brook, N.Y. 11794

ABSTRACT

Although a great deal of research effort has been expended in support of natural language (NL) database querying, little effort has gone to NL database update. One reason for this state of affairs is that in NL querying, one can tie nouns and stative verbs in the query to database objects (relation names, attributes and domain values). In many cases this correspondence seems sufficient to interpret NL queries. NL update seems to require database counterparts for active verbs, such as "hire," "schedule" and "enroll," in addition to what is needed for NL querying. There currently seems to be no natural candidate to fill this role.

We suggest a database counterpart for active verbs, which we call verbgraphs. The verbgraphs may be used to support NL update. A verbgraph is a structure for representing the various database changes that a given verb might describe. In addition to describing the variants of a verb, it may be used to disambiguate the update command. Other possible uses of verbgraphs include specification of defaults, prompting of the user to guide but not dictate user interaction, and enforcing database integrity constraints.

I. MOTIVIATION AND PROBLEM STATEMENT

We want to support natural language interface for all aspects of database manipulation. English and English-like query systems alrendy exist, such as ROBOT[Ha77], TQA[Da78], LUNAR[WO76] and those described by Kaplan[Ka79], Walker[Wa78] and Waltz [Wz75]. We propose to extend natural language interaction to include data modification (insert, delete, modify), rather than simply data extraction. The desirability and unavailability of natural language database modification has been noted by Wiederhold, et al.[Wi81]. Database systems currently do not contain structures for explicit modelling of real world changes.

A database (DB) is an attempt to abstract information about the real world. A state of the DB is meant to represent a state of a portion of the real world. We refer to the abstract description of the portion of the real world being modelled as the semantic data description (SDD). A SDD indicates a set of real world states (RWS) of interest, a DB definition gives a set of allowable database states (DBS). The correspondence between the SDD and the DB definition induces connections between DB states and real world states. The situation is diagrammed in Figure 1.

Natural Language (NL) querying of the DB requires that the correspondence between the semantic description and the DB definition be explicitly stated. The query system must translate a question phrased in terms of the semantic description into a question phrased as a data retrieval command in the language of the DB system. The response to the command must be translated back into terms of the SDD, which yields information about the real world state. For NL database modification, this stative correspondence between DB states and real world states is not adequate. We want changes in the real world to be reflected in the DB. In Figure 2 we see that when some action in the real world causes a state change from RWS1 to RWS2, we must perform some modification to the DB to change its state from DBS1 to DBS2.

We have a means to describe the action that changed the state of the real world: active verbs. We also have a means to describe a change in the DB state: a data manipulation language (DML) command sequence. But given a real world action, how do we find a DML command sequence that will accomplish the corresponding change in the DB?

Before we explore ways to represent this active correspondence--the connection between real world actions and DB updates--, let us examine how the stative correspondence is captured for use by a NL query system. We need to connect entities and relationships in the semantic description with files, fields and field values in the DB. This stative correspondence between RWS and DBS is generally specified in a system file. For example, in Harris' ROBOT system, the semantic description is implicit, and it is assumed to be given in English. The entities and relationships in the description are roughly

*This research is partially supported by NSF grants IST-79-18264 and ENG-79-07994.

English nouns and stative verbs. The correspondence of the semantic description to the DB is given by a lexicon that associates English words with files, fields and field values in the DB. This lexicon also gives possible referents for word and phrases such as "who," "where" and "how much."

Consider the following example. Suppose we have an office DB of employees and their scheduled meetings, reservations for meeting rooms and messages from one employee to another. We capture this information in the following four relations:

EMP(name, office, phone, supervisor)
APPOINTMENT(name, date, time, duration, who, topic, location)
MAILBOX(name, date, time, from, message)
ROOMRESERVE(room, date, time, duration, reserver)

with domains (permissible sets of values):

DOMAIN	ATTRIBUTES WITH THAT DOMAIN
personname	name, who, from, reserver, supervisor
roomnum	room, location, office
phonenum	phone
calendardate	date
clocktime	time
elapsedtime	duration
text	message, topic.

Consider an analysis of the query

"What is the name and phone # of the person who reserved room 85 for 2:45pm today?"

Using the lexicon, we can tie words in the query to domains and relations

name - personname
phone - phonenum
person - personname
who - personname
reserved - ROOMRESERVE relation
room - roomnum
2:45pm - clocktime
today - calendardate

We need to connect relations EMP and ROOMRESERVE. The possible joins are room-office and name-reserver. If we have stored the information that offices and reservable rooms never intersect, we can eliminate the first possibility. Thus we can arrive at the query

in EMP, ROOMRESERVE retrieve name, phone where name=reserver and room=85 and time=2:45pm and date=CURRENTDATE

(We assume we have access to some system maintained variables such as CURRENTTIME and CURRENTDATE.)

Suppose we now want to make a change to the database:

"Schedule Bob Marley for 2:15pm Friday."

This request could mean schedule a meeting with an individual or schedule Bob Marley for a seminar. We want to connect "schedule" with the insertion of a tuple in either APPOINTMENT or ROOMRESERVE. Although we may have pointers from "schedule" to APPOINTMENT and ROOMRESERVE, we do not have adequate information for choosing the relation to update. We need a way to generate a question that will distinguish the two senses. Then additional information must be requested to construct a complete tuple for insertion. Finally, there may be integrity constraints to check, such as no one can have two appointments at once, or the same room cannot be reserved twice at the same time.

Although files, fields, domains and values seem to be adequate for expressing the stative correspondence, we have no similar DB objects to which we may tie verbs that describe actions in the real world. The best we can do with files, fields and domains is to indicate what is to be modified; we cannot specify how to make the modification. We need to connect the verbs "schedule," "hire" and "reserve" with some structures that dictate appropriate DML sequences to perform the corresponding updates to the DB. In addition, we have seen that a verb may denote various actions, that is, it may have different senses. The particular sense can depend on the entities involved in the action. There can also be variants within a given sense. To what can "schedule" be connected to in the DB? There is no explicit database object that represents all the changes in the database that correspond to the changes in the real world brought about by the action "schedule." The best we have is a specific DML command sequence, a transaction, for each instance of "schedule" in the real world. No single transaction truly represents all the implications and variants of the "schedule" action. "Schedule" really corresponds to a set of similar transactions, or perhaps some parameterized version of a DB transaction. However, there is no such "parameterized transaction" in the DB with which to connect "schedule." Our approach is to design and employ a structure on the DB side that explicitly represents parameterized transactions.

The desired situation is shown in Figure 3, where RWS1 statively corresponds to DBS1. We have an active correspondence between "schedule" and a parameterized DB transaction (PT). Different instances of the schedule action, S1 and S2, cause different changes in the real world state, from RWS1 to RWS2 or to RWS3. From the active correspondence of "schedule" and the PT, we want to produce the proper transaction, T1 or T2, to effect the correct change in the DB state.

To implement the parameterized transaction outlined above, which issues a correct DML sequence for a verb in the real world, someone could, of course, write a separate program for each action to generate the DML commands. We reject this approach because we want a higher-level and more structured representation. We desire a high-level description for updates that corresponds to verb senses and their variants. We want a system that, given a description of an action in the real world using that verb sense, will automatically select a DML sequence that properly updates the database.

We have additional uses in mind for these higher-level descriptions. First, they can help us select from among senses of a given root verb used in an action description. Another is to specify the necessary information the action description must contain to properly select a DML sequence. It should also serve as a convenient means to specify defaults. The verb representation can also serve to express constraints on the update operation, just as functional dependencies represent constraints on the state of a database.

What must the high-level verb descriptions look like and how should a system that uses them operate? We must be able to readily express the correspondence between actions in the semantic world and verb descriptions in this high-level specification. We depend heavily on this correspondence to process natural language updates, just as the stative correspondence is used to process natural language queries. Finally, the verb description helps to disambiguate multiple verb senses and aids in selection of the proper variant of a given verb sense. In the next section we examine these requirements in more detail and offer, by example, one candidate for the representation.

Another indication that active verbs are a problem in DBs shows up in a semantic data models. Semantic data models are systems for constructing precise descriptions of portions of the real world - semantic data descriptions (SDD) - using terms that come from the real world rather than a particular DB system. A SDD is a starting point for designing and comparing particular DB implementations. Some of the semantic models that have been proposed are the entity-relationship model[Ch76], SDM[HM81], RM/T[Co79], and Beta[Br78]. For some of these models, methodologies exist for translating to a DB specification in various DB models, as well as for expressing the static correspondence between a SDD in the semantic model and a particular DB implementation. These models generally have constructs corresponding to entities, attributes and relationships (stative verbs) in the real world, which can be given natural names: person, height, supervises. To express actions in these models, however, there are only terms that refer to DBs: insert, delete, modify, rather than schedule, cancel, postpone (the notable exceptions are Skuce[Sk80]and TAXIS[MBW80]). Such terms are not useful for expressing the correspondence of real world actions to DB changes, since they already refer to the DB. Perhaps the deficiency exists because of the difficulty of extending the translation methodology to actions, or even expressing the correspondence between action and database modification.

While there have been a number of approaches made to NL querying, there seems to be little work on NL update. Carbonell and Hayes[CH81] have looked at parsing a limited set of NL update commands, but they do not say much about generating the DB transactions for these commands. Kaplan and Davidson [KD81] have looked at the translation of NL updates to transactions, but the active verbs they deal with are synonyms for DB terms, essentially following the semantic data models as above. This limitation is intentional, as the following excerpt shows:

> First, it is assumed that the underlying database update must be a series of transactions of the same type indicated in the request. That is, if the update requests a deletion, this can only be mapped into a series of deletions in the database.

While some active verbs, such as "schedule," may correspond to a single type of DB update, there are other verbs that will require multiple types of DB updates, such as "cancel," which might require sending message as well as removing an appointment. Kaplan and Davidson are also trying to be domain independent, while we are trying to exploit domain-specific information.

II. NATURE OF THE REPRESENTATION

We propose a structure, a verbgraph, to represent action verbs. Verbgraphs are extensions of frame-like structures used to represent verb meaning in MORAN[Sa78, Sa79]. One verbgraph is associated with each sense of a verb; that structure represents all variants. A real world change is described by a sentence that contains an active verb; the DB changes are accomplished by DML command sequences.

A verbgraph is used to select DML sequences appropriate to process the variants of a verb sense. The primitives in these structures are the relations, attributes and values from the DB, employed in DML-like expressions. We actually generate transactions in some intermediate language (IL), rather than a particular DML, in order to avoid trying the representation to a particular database system. The IL can then be translated into various DMLs. We also wish to capture that one verb may be used as part of another. An analog is subparts in the noun world, where month, day and year may be subparts of date. Similarly, we may have a verb sense RESERVE-ROOM that may be used by itself or may be used as a subpart of the verb SCHEDULE-TALK. We want our representation structure to capture the knowledge that some IL command sequences may be repeatedly used as subparts of other, larger IL command sequences.

Figure 4 is an example of one possible structure for a verbgraph. It models the "schedule appointment" sense of the verb "schedule." There are four basic variants we are attempting to capture; they are distinguished by whether or not the appointment is scheduled with someone in the company and whether or not a meeting room is to be reserved. There is also the possibility that the supervisor must be notified of the meeting. The different operations for each variant are described below. In every case the person scheduling the appointment gets an entry in the APPOINTMENT relation.

1) Meeting with person in company, no reserved room. Make an appointment entry for the other person and leave a message. The location of the meeting will be one of the two offices of the people involved. Optionally, supervisor may be notified.

2) Meeting with person not in company, no reserved room. Meeting will be in scheduler's office. Optionally, supervisor notified.

3) Meeting with another employee, reserve a room. Make an appointment entry for the other person and send a message. Reserve a meeting room for the same time. Optionally, supervisor may be notified.

4) Meeting with person not in company, reserve a room. Reserve a meeting room for the same time. Supervisor must be notified.

The verbgraph is directed acyclic graph (DAG) with 5 kinds of nodes: header, footer, information, AND ⊕ and OR ○ . Header is the source of the graph, the footer is the sink. Every information node has one incoming and outgoing edge. An AND or OR node can have any number of incoming or outgoing edges. A variant corresponds to a directed path in the graph. We define a path to be connected subgraph such that 1) the header is included; 2) the footer is included; 3) if it contains an information node, it contains the incoming and outgoing edge; 4) if it contains an AND node, it contains all incoming and outgoing edges; and 5) if it contains an OR node, it contains exactly one incoming and one outgoing edge. We can think of tracing a path in the graph by starting at the header and following its outgoing edge. Whenever we encounter an information node, we go through it. Whenever we encounter an AND node, the path divides and follows all outgoing edges. We may only pass through an AND node if all its incoming edges have been followed. An OR node can be entered on only one edge and we leave it by any of its outgoing edges.

An example of a complete path is one that consists of the header, footer, information nodes, A, B, D, J, and connector nodes a, b, c, d, g, k, l, n. Although there is a direction to paths we do not intend that the order of nodes on a path implies any order of processing the graph, except the footer node is always last to be processed. A variant of a verb sense is described by the set of all expressions in the information nodes contained in a path.

Expressions in the information nodes can be of two basic types: assignment and restriction. The assignment type produces a value to be used in the update, either by input or computation; the key word input indicates the value comes from the user. Some examples of assignment are:

1) (node labelled A in Figure 4) APPT.who ← input from personname

The user must provide a value from the domain personname.

2) (node labelled D in Figure 4) RES.date ← APPT.date

The value for APPT.date is used as the value RES.date.

The form of a restriction is
<tvar>.<attrname> (not) in <valueset>.

An example of restriction is: (node B in Figure 4) APPT.who in R1 where R1 = in EMP retrieve name .

These statements restrict the value of APPT.who to either be a company employee or not. Also in Figure 4, the symbols R_1, R_2, R_3 and R_4 stand for the retrievals

R_1 = in EMP retrieve name
R_2 = in EMP retrieve office where name = APPT.name
R_3 = in EMP retrieve office where name = APPT.name or name = APPT.who.
R_4 = in EMP retrieve supervisor where name APPT.name.

In Node B, INFORM(APPT.who, APPT.name, 'meeting with me on %APPT.date at %APPT.time') stands for another verbgraph that represents sending a message by inserting a tuple in MAILBOX. We can treat the INFORM verbgraph as a procedure by specifying values for all slots that must be filled from input. The input slots for INFORM are (name, from, message).

III. WHAT CAN WE DO WITH IT?

One use for the verbgraphs is in support of NL directed manipulation of the DB. In particular, they can aid in variant selection. We assume that correct verb sense has already been selected; we discuss sense selection later. Our goal is to use information in the query and user responses to questions to identify a path in the verbgraph. Let us refer again to the verbgraph for SCHEDULE-APPOINTMENT shown in Figure 4. Suppose the user command is "Schedule appointment with James Parker on April 13" where James Parker is a company employee. Interaction with the verbgraph proceeds as follows. First, information is extracted from the command and classified by domain. For example, James Parker is in domain personname, which can only be used to instantiate APPT.name, APPT.who, APPT2.name and APPT2.who. However, since USER is a system variable, the only slots left are APPT.who and APPT2.name, which are necessarily the same. Thus we can instantiate APPT.who and APPT2.name with "James Parker." We classify "April 13" as a calendar-date and instantiate APPT.date, APPT2.date and RES.date with it, because all these must be the same. No more useful information is in the query.

Second, we examine the graph to see if a unique path has been determined. In this case it has not. However, other possibilities are constrained because we know the path must go through node B. This is because the path must go through either node B or node C and by analyzing the response to retrieval R1, we can determine it must be node B (i.e., James Parker is a company employee).

Now we must determine the rest of the path. One determination yet to be made is whether or not node D is in the path. Because no room was mentioned in the query, we generate from the graph a question such as "Where will the appointment take place?" Suppose the answer is "my office." Presume we can translate "my office" into the scheduler's office number. This response has two effects. First, we know that no room has to be reserved, so node D is not in the

path. Second, we can fill APPT.where in node F. Finally, all that remains to be decided is if node H is on the path. A question like "Should we notify your supervisor?" is generated. Supposing the answer is "no." Now the path is completely determined; it contains nodes A, B and F.

Now that we have determined a unique path in the graph, we discover that not all the information has been filled-in in every node on the path. We now ask questions to complete these nodes, such as "What time?", "For how long?" and "What is the topic?". At this point we have a complete unique path, so the appropriate calls to INFORM can be made and the parameterized IL in the footer can be filled in.

Note that the above interaction was quite rigidly structured. In particular after the user issues the original command, the verbgraph instantiation program chooses the order of the subsequent data entry. There is no provision for default, or optional values. Even if optional values were allowed, the program would have to ask questions for them anyway, since the user has no opportunity to specify them subsequent to the original command. We want the interaction to be more user-directed. Our general principle is to allow the user to volunteer additional information during the course of the interaction, as long as the path has not been determined and values remain unspecified. We use the following interaction protocol. The user enters the initial command and hits return. The program will accept additional lines of input. However, if the user just hits return, and the program needs more information, the program will generate a question. The user then answers that question, followed by a return. As before, additional information may be entered on subsequent lines. If the user hits return on an empty line, another question is generated, if necessary.

The following advantages accure from letting the user volunteer information. The user may choose the order of data entry. We can now have optional values, but not have to ask questions about them. Since the user has an opportunity to volunteer any values, if he or she does not volunteer the value, a default value will be used.

Brodie[Br81] and Skuce[Sk80] both present systems for representing DB change. Skuce's goal is to provide an English-like syntax for DB procedure specification. Procedures have a rigid format and require all information to be entered at the time of invocation in a specific order, as with any computer subprogram. Brodie is attempting to also specify DB procedures for DB change. He allows some information to be specified later, but the order is fixed. He also gets information from the DB when possible. Neither allow the user to choose the order of entry, and neither accomodates variants that would require different sets of values to be specified. However, like our method, and unlike Kaplan and Davidson [KD81], they attempt to model DB changes that correspond to real world actions rather than just specifying English synonyms for single DB commands.

We are currently considering hierarchically structured transactions, as used in the TAXIS semantic model [MBW80], as an alternative to verbgraphs. Verbgraphs can be ambiguous, and do not lend themselves to top-down design. Hierarchical transactions would seem to overcome both problems. Hierarchical transactions in TAXIS are not quite as versatile as verbgraphs in representing variants. The hierarchy is induced by hierarchies on the entities classes involved. Variants based on the relationship among particular entities, as recorded in the database, cannot be represented. Also all variants in the hierarchy must involve the same entity classes, where we may want to involve some classes only in certain variants. However, these shortcomings do not seem insurmountable.

Certain constraints on updates are implicit on verbgraphs, such as APPT.where + input from R_3, which constrains the location of the meeting to be the office of one of the two employees. We can also use verbgraphs to maintain database consistency. Integrity constraints take two forms: constraints on a single state and constraints on successive database states. The second kind is harder to enforce; few systems support constraints on successive states. There are also constraints on successive database states particular to a given update action. For example, if we had a verbgraph for postponing appointments, it should check that the new appointment time is later than the current appointment time, although this is not a general constraint on changing appointments.

Verbgraphs provide many opportunities for specifying various defaults. First, we can specify default values, which may depend on other values. Second, we can specify default paths. Verbgraphs are also a means for specifying non-DB operations. For example, if an appointment is made with someone outside the company, generate a confirmation letter to be sent.

All of the above discussion has assumed we are selecting a variant where the sense has already been determined. In general sense selection, being equivalent to the frame selection problem in Artifical Intelligence[CW76], is very difficult. We do feel that verbgraph will aid in sense selection, but will not be as efficacious as for variant selection. In such a situation, perhaps the English parser can help disambiguate or we may want to ask an appropriate question to select the correct sense, or as a last resort, provide menu selection.

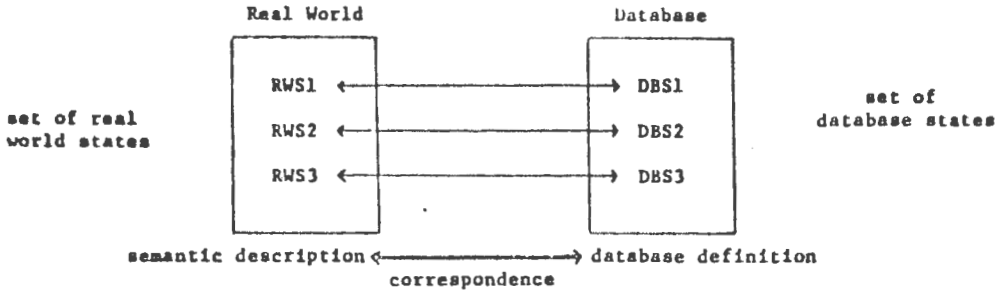


FIGURE 1

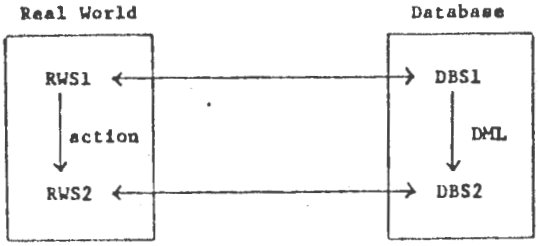


FIGURE 2

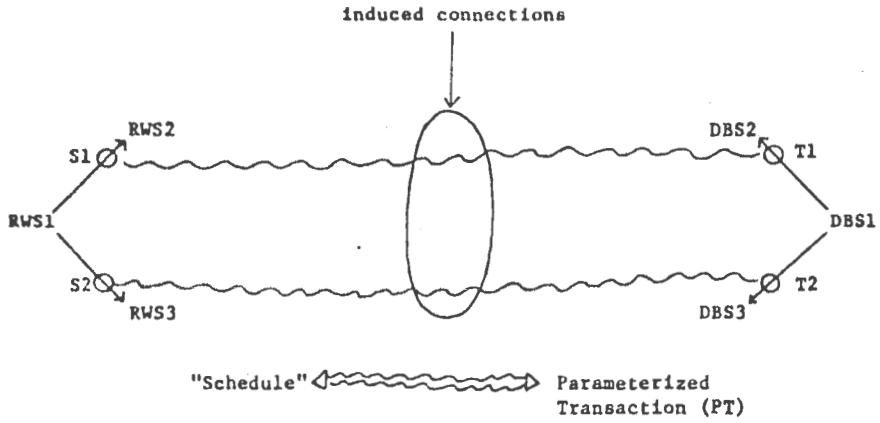


FIGURE 3

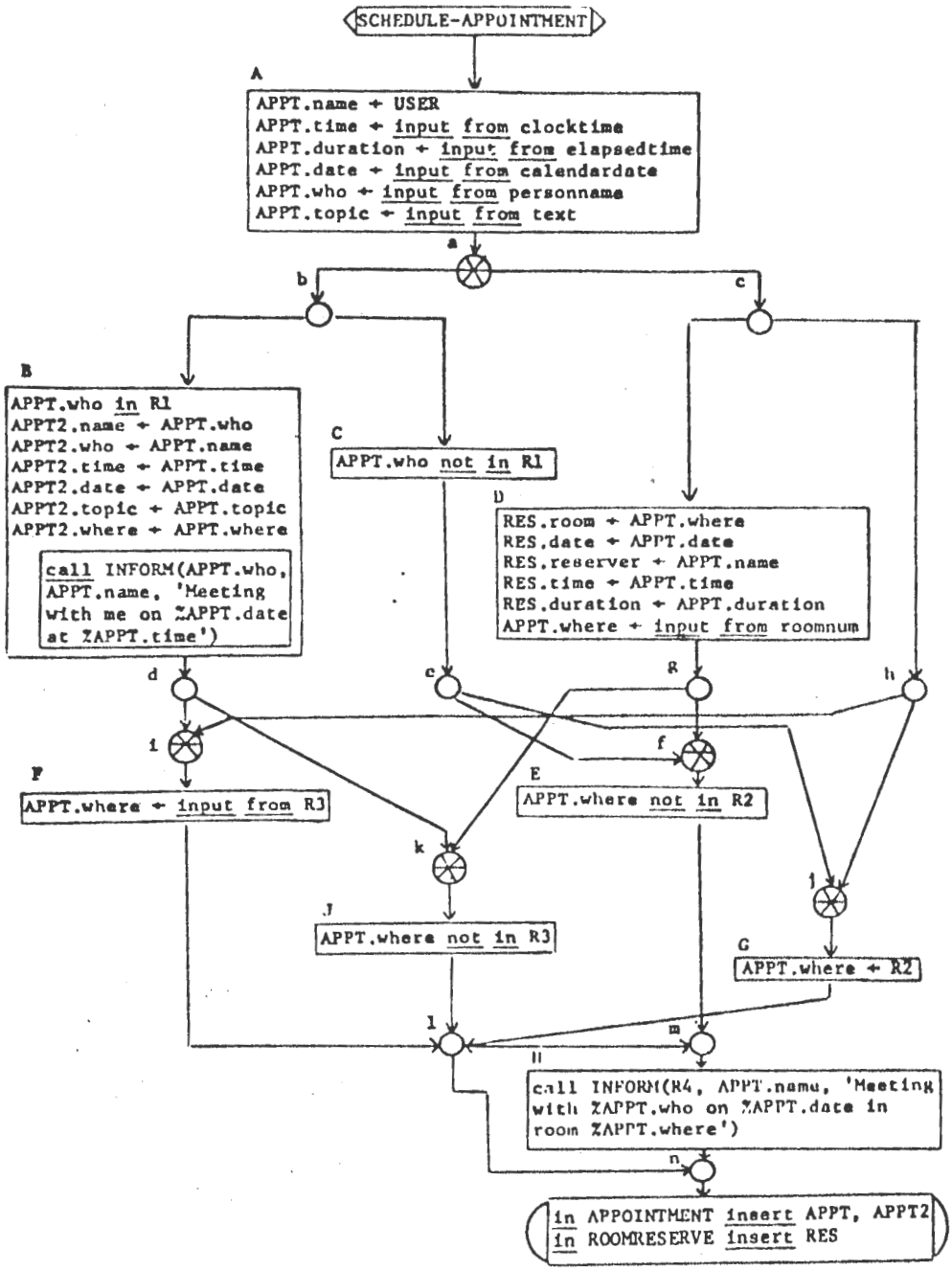

SCHEDULE-APPOINTMENT
A
APPT.name ← USER
APPT.time ← input from clocktime
APPT.duration ← input from elapsedtime
APPT.date ← input from calendardate
APPT.who ← input from personname
APPT.topic ← input from text
a
b
c
B
APPT.who in R1
APPT2.name ← APPT.who
APPT2.who ← APPT.name
APPT2.time ← APPT.time
APPT2.date ← APPT.date
APPT2.topic ← APPT.topic
APPT2.where ← APPT.where
call INFORM(APPT.who, APPT.name, 'Meeting with me on %APPT.date at %APPT.time')
C
APPT.who not in R1
D
RES.room ← APPT.where
RES.date ← APPT.date
RES.reserver ← APPT.name
RES.time ← APPT.time
RES.duration ← APPT.duration
APPT.where ← input from roomnum
d
e
g
h
i
f
F
APPT.where ← input from R3
E
APPT.where not in R2
k
j
J
APPT.where not in R3
G
APPT.where ← R2
l
m
H
call INFORM(R4, APPT.name, 'Meeting with %APPT.who on %APPT.date in room %APPT.where')
n
in APPOINTMENT insert APPT, APPT2
in ROOMRESERVE insert RES

FIGURE 4

REFERENCES

[Br78] Brodie, N.L., Specification and verification of data base semantic integrity. CSRG Report 91, Univ. of Toronto, April 1978.

[Br81] Brodie, M.L., On modelling behavioral semantics of database. VLDB 7, Cannes, France, Sept. 1981.

[CH81] Carbonell, J. and Hayes, P., multi-strategy construction-specification parsing for flexible database query and update. CMU Internal Report, July 1981.

[CW76] Chen, P. P.-S., The entity-relationship model: toward a unified view of data. ACM TODS 1:1, March 1976, pp. 9-36.

[Co79] Codd, E.F., Extending the database relational model to capture more meaning. ACM TODS 4:4, December 1979, pp. 397-434.

[Da78] Damereau, F.J., The derivation of answers from logical forms in a question answering system. American Journal of Computational Linguistics, Microfiche 75, 1978, pp.3-42.

[HM81] Hammer, M. and McLeod, D., Database description with SDM: A semantic database model. ACM TODS 6:3, Sept. 1981, pp. 351-386.

[Ha77] Harris, L.R., Using the database itself as a semantic component to aid the parsing of natural language data base queries. Dartmouth College Mathematics Dept. TR 77-2, 1977.

[Ka79] Kaplan, S.J., Cooperative responses from a natural language data base query system. Stanford Univ. Heuristic Programming Project paper HPP-79-19.

[KD81] Kaplan, S.J., and Davidson, J., Interpreting Natural Language Updates-proceedings of the 19th Annual Meeting of the Association for Computational Linguistics, June 1981.

[MBW80] Mylopoulos, J., Bernstein, P.A. and Wong, H.K.T., A language facility for designing database-intensive applications. ACM TODS 5, 2, June 1980, pp. 185-207.

[Sa78] Salveter, S.C., Inferring conceptual structures from pictorial input data. University of Wisconsin, Computer Science Dept., TR 328, 1978.

[Sa79] Salveter, S.C., Inferring conceptual graphs. Cognitive Science, 3, pp. 141-166.

[Sk80] Skuce, D.R., Bridging the gap between natural and computer language. Proc. of Int'l Congress on Applied Systems, and Cybernetics, Acapulco, December 1980.

[Wa78] Walker, D.E., Understanding Spoken Language. American Elsevier, 1978.

[Wi81] Wiederhold, G., Kaplan, S.J. and Sagalowicz, D., Research in knowledge base management systems. SIGMOD Record, VII, #3, April 1981, pp. 26-54.

[Wo76] Woods, W., et. al., Speech Understanding Systems: Final Technical Progress Report, BBN No. 3438, Cambridge, MA, 1976.

[Wz75] Waltz, D., Natural language access to a large database: an engineering approach. In Proc. of the Fourth Int'l Joint Conf. on Artifical Intelligence, 1976.

Natural Language Access to Databases: User Modeling and Focus

Jim Davidson
Computer Science Department
Stanford University
Stanford, California 94305

Abstract

Users of natural language database systems will sometimes phrase inputs with respect to the perceived *focus* of the dialogue. In order to interpret such inputs correctly, the system must retain a model of the user's current focus. This paper discusses the requirements of such a model, and describes the method of user modeling implemented in the PIQUE system. This method relies on retention of a formal description of the segment of the database currently in focus. A number of examples of the use of the focus model are presented, and its applicability and limitations are discussed.

1. Introduction

Natural language database systems which do not retain user models will behave inappropriately in a large class of naturally occurring situations. Without such a model, the system may misinterpret the user's input, mis-handling such linguistic phenomena as definite noun phrase reference, word-sense ambiguity, and grammatical ambiguity.

Consider the following dialogue between a user and a database management system[1]:

Q1: Who are the programmers?
R1: Jones, Smith, Baker
Q2: What is Jones' salary?
R2: There are 37 employees named "Jones";
which one do you mean?

The system here is being uncooperative, failing to recognize the (apparent) intent of the user's second query. This problem occurs because the system attempts to interpret the query in isolation; on this basis, the referring noun phrase "Jones" in Q2 is genuinely ambiguous. However, the context of the first Q/R pair strongly suggests a likely referent, and this should be detected by the system.

[1] example due to Bob Moore

The use of the abbreviated form of reference by the user for his second request is not an isolated occurrence. Users of "intelligent" systems will tend to attribute human-like intelligence to those systems. In the case of natural language systems, this means that the users will observe some of the rules of conversational coherence, and phrase inputs with respect to the current context. In the example above, the user has obeyed the Cooperative Principle (Grice, 1975), and made his specification exactly as informative as (he believes to be) necessary. To prevent the kind of failure which has occurred here, the system must retain some model, however simple, of the user's current "state of mind."

This paper describes a simple yet adequate approach to user modeling, which has been implemented as part of the *PIQUE* (Program for Interpretation of Queries and Updates in English) natural language database system (Kaplan and Davidson, 1981)[2]. This approach is derived from formal work in databases, and relies on retention of induced *views*, which are analogous to the view mechanism in database management. This method allows the system to interpret correctly the phenomenon discussed above (and others), but does not require any additional linguistic capabilities in the natural language interface, nor any additional domain-dependent information beyond that encoded in the database schema and the database itself.

The next section contains a discussion of user modeling as it has been done in artificial intelligence and in database management. Section 3 describes, in detail, the approach to user modeling used in PIQUE. Section 4 contains a detailed presentation of the use of this approach to address one problem: interpretation of definite noun phrase reference. The following section illustrates a number of other problems which are amenable to the focus model approach. The final

[2] Examples similar to the ones presented in the paper have been run on the PIQUE system. PIQUE is written in INTERLISP, and runs on the DECsystem-20 at SRI. The PIQUE parser is written in LIFER (Hendrix, 1977)

section contains a discussion of the coverage and adequacy of this method.

2. User modeling in artificial intelligence and database management

Classes of user models for computer programs can be divided along three dimensions (Rich, 1979): *explicit/implicit* (does the user have to build the model himself, or does the system infer it from his behavior), *canonical/individual* (is the model intended to represent users in general, or is there a different model for each user), and *long-term/short-term* (is the model intended to represent the user's profile in general, or is it intended to be more dynamic, changing as he focuses on different tasks). Within this framework, the class of models considered here is implicit, individual, and short-term. A better term than "user model" for this class might be "focus model", since it concerns the user's current interest, rather than his global characteristics; we will use the latter term.

There has been some work in artificial intelligence on the use of focus models in dialogues, although none in the context of database access. Grosz (1977) developed a system for modeling focus in task-oriented dialogues. That model relied on a domain-specific representation of the *task hierarchy*—the global relationship between the tasks and sub-tasks being worked on. The focus model (which was represented using partitioned semantic networks) was used mainly to resolve non-pronominal definite noun phrases appearing in the discourse. Sidner (1979) described a more general method of anaphora comprehension, which relied on a model of focus similar to Grosz's. Cohen and Perrault (1979) developed a sophisticated model, capable of representing the user's wants, beliefs, and intentions, as part of a plan-based theory of speech acts.

Database interactions present particular problems for focus modeling, because there is, in general, no *a priori* structure to the kinds of dialogues which can occur (as there is, for example, in task-oriented dialogues). The user's access of the database will follow no global pattern.

Within database management, only restricted forms of user modeling have been provided. Database theory provides a formal notion of an *external model* (also called data submodel). The external model is a transformation of the conceptual model (the "actual" database), representing the aspect of the database visible to a user or group of users. The external model is composed of *views* (derived relations), that are formed from the underlying database by operations such as *projection* (e.g., a user may be allowed to examine employee records, but not the salary attribute) or *restriction* (e.g., the user may examine the records of only those employees in the sales department). However, the structure of the external model is long-term, and must be fully specified in advance of any transactions involving the model, by the database administrator.

Short-term models have been considered in only a few instances. Rowe (1982) has implemented a system that takes into account the user's *preferences* (as dictated by the task that he is trying to accomplish) to decide which items from the database to present, and in which order to present them. However, this depends upon (a) having a pre-specified model of the task to be performed, and (b) recognizing the current dialogue as an instance of an attempt to perform that task. Finkelstein (1982a) uses a short-term, implicit, individual model, to improve efficiency of query response by recognizing commonalities between successive queries. While his mechanism is similar to ours, his goal—optimization of response time—is very different from ours, which might be termed *cooperativeness* or *habitability*.

3. A formal mechanism for representing focus in database systems

This section presents a focus representation which has been developed as part of the PIQUE system, together with the motivation for this model. In brief, the PIQUE focus representation models the user by noting which aspects of the database he has previously examined.

The development of a focus model entails three considerations: (a) a representation for the user's current focus; (b) a method of *deriving* the focus representation, and maintaining the model during the course of a dialogue; (c) an explicit mechanism for *using* the focus representation, when appropriate, in interpretation, or response generation. The representation and derivation of the focus will be detailed here. The *use* of the representation, since it varies with the type of the problem, will be explained together with the examples in the following two sections.

3.a) The focus representation

The user's queries to the database are expressed, at some level, in a formal data manipulation language (DML), which is typically a variant of the relational calculus or relational

algebra (Ullman, 1980). For example, the query "Who are the programmers?" might be expressed, in an idealized calculus-based DML:

```
{ x.name : x∈emps | x.occupation="programmer" }
```

(i.e., "Print the names of all the members of the employees relation whose occupation is 'programmer'.")

This expression can be viewed as an *intensional description* of the class of programmers, as well as as a query. Thus, the DML can serve as a representation language for describing segments of the database (a representation language that has the benefit of a formal semantics). The user's focus is assumed to be that segment of the database that he is currently accessing, so in PIQUE, the current focus, at any time, is represented intensionally by a DML expression. The intension (description) is a more useful concept than the *extension* because it describes, not only the entities that are currently in the focus, but also the *aspect* of them that is currently of interest.

Focus expressions encoded in the DML bear some resemblance to *views*, as described in the previous section; both are described with DML expressions. The difference is that focus is short-term and implicit, whereas views are long-term, and explicitly specified by the database administrator.

3. b) Deriving the focus representation from the dialogue

PIQUE uses a simple method of tracking the focus in a dialogue. Each request by the user establishes a temporary "focus space", represented by the DML form of the request. Successive inputs may make use of this space, or shift to a new focus space. Focus spaces are "stacked", to allow reference back to events or entities mentioned farther back in the dialogue. Such references are rare, however.

Unlike Grosz's work, the database domain does not provide strong clues for the closing of a focus space. Rather, such action is indicated by a shift to a new focus space; shifts are indicated by queries which examine different areas of the database, queries which invoke entities outside the previous focus space, etc.

[3] The actual DML used in PIQUE is a modified form of SODA, a LISP-compatible variant of relational calculus, developed by Bob Moore; except for syntax, it is fundamentally identical to the DML used here.

4. Use of focus in interpreting non-pronominal definite noun phrases

A focus space, in general, establishes a "highlighted" subset of the objects or entities in the domain of discourse. In the database context, the entities of the domain are those that appear as entities in the database ("entity" is used here in the sense of Chen (1976).)

The focus space identifies subsets of certain entity sets. This focus space might then provide the referent for referring expressions in subsequent dialogue. Further queries which reference the entity set in question may be evaluated over the subset, instead of the entire set. The interpretation-in-context can be effected via *query modification*: the DML expression of the subsequent query can be modified algorithmically to incorporate the intension of the focus. (See, for example, the *INGRES* system (Stonebraker, 1975).)

Consider again the example of Section 1. The focus representation, established by Q1, "Who are the programmers?", would be expressed in the DML as:

```
{ x.name : x∈emps | x.occupation="programmer" }
```

identifying a subset of the set of employees.

The initial interpretation of Q2, "What is Jones' salary?", (without context) would be:

```
{ x.sal : x∈emps | x.name="Jones" }
```

Q2 may be modified, to range over the subset established in Q1:

```
{ x.sal : x∈emps | x.name="Jones"∧
                   x.occupation="programmer" }
```

The decision to use the focus in interpreting a query is based on a number of factors, and is discussed below.

Consider a more complex example of definite noun phrase reference:

Q1: List the name and type of all American ships that are docked in French ports.

R1:

Name	type
Kranj	supertanker
Totor	tanker
Pequod	bulk carrier
.....	

Q2: What cargoes are the tankers carrying?

The DML corresponding to Q1 is:

```
{ x.name, x.type : x∈ships, y∈ports |
  y.portnm=x.portnm ∧ y.country="France" ∧
  x.country="US" }
```

(i.e., containing a single *join* between the *ships* and *ports* relations), and the DML for the global interpretation for Q2 is:

```
{ y.cargo : x∈ships, y∈shipments |
  x.type="tanker" ∧ y.shipment#=x.shipment# }
```

(also containing a single join; note that in the database, *shipments* are kept in a separate relation, to preserve a permanent record).

The focus determines a subset of the *ships* relation; Q2 can be interpreted with respect to that subset, to mean, "What cargoes are American tankers that are docked in French ports carrying?":

```
{ y.cargo : x∈ships, y∈shipments, z∈ports |
  x.type="tanker" ∧ x.country="US" ∧
  y.shipment#=x.shipment# ∧ z.portnm=x.portnm
  ∧ z.country="France"       }
```

(Note that renaming of variables may be necessary during query modification.)

The focus space also induces a subset of the *ports* relation. If Q2 had been "What are the names of the ports?",

```
{ x.portnm : x∈ports }
```

a reasonable interpretation would have been to interpret it with respect to the focus space, to mean, "What are the names of French ports that have American ships docked in them?":

```
{ x.portnm : x∈ports, y∈ships |
  y.portnm=x.portnm ∧ x.country="France" ∧
  y.country="US" }
```

In natural dialogue, items not explicitly mentioned may sometimes be considered to be "in focus". Consider, "I bought a new briefcase yesterday, and the handle broke". The phrase "the handle", although not mentioned previously, is in focus through a type of *foregrounding*. This phenomenon of *implicit focus* arises occasionally in natural language querying; consider a database of projects and parts, where parts have numbers, costs, etc.:

Q1: What parts are needed for project 10?
R1: d-12, j-79, ...
Q2: What are the costs?

Clearly, the costs requested are those of the parts mentioned in R1. However, Q2 contains nothing to relate it to Q1. The difference between this example and the previous ones is that in the earlier cases the follow-up query explicitly mentioned an attribute or relation that appeared in the focus space (or a value for such an attribute).

This phenomenon can be handled with the same mechanism. The initial interpretation of Q2 is:

```
{ x.cost : x∈part }
```

(i.e., list all the costs in the database). This interpretation is dictated by the database schema, which finds that costs occur only as an attribute of parts. Since the variable of Q2 ranges over the "parts" relation, and the focus space provides a restriction on that relation, the query may be interpreted in context. Thus, the database schema and focus mechanism, together, provide a type of implicit focus.

4.a) Appropriateness and efficacy

Like all heuristic approaches, the method discussed here has limitations. In this section, we consider the class of situations in which the model is appropriate, the kinds of errors that can arise in the use of the model, and methods of avoiding inappropriate use.

For the focus model discussed here, an *inappropriate* effect would be to evaluate the query with respect to a restricted focus space, when the user had not intended this restriction. It is instructive to consider the kinds of errors which can arise from inappropriate application of the focus mechanism.

Consider the sequence:

Q1: Which employees make more than $20K?
R1: Forsythe, King ....
Q2: Who lives in San Francisco?

Suppose that the system assumes that the second query refers to employees making more than $20K who live in San Francisco, when this is not actually the case. The answer returned from Q2 will be *incomplete*—omitting many employees—but not *wrong*: all the information that does appear will be correct.

As another example, consider:

Q1: Which suppliers are located in LA?
R1: ABC, ...
Q2: Which parts do not have suppliers?

If the restricted version of "suppliers" induced by the first query is used in interpreting the second query, some parts

might erroneously be included in the answer set (i.e., parts which have only suppliers outside of L.A.).

The difference between these two queries is that the first is *monotonic* (Finkelstein, 1982b). Roughly, monotonic queries are those which contain neither *universal quantification*, nor (certain forms of) *negation*. (In the relational algebra, these correspond to the operations of *division* and *set difference*.) This is a large class of queries, which includes the common *select-project-join* queries. Only non-monotonic queries such as Q2 in the second example above admit errors of incorrectness. PIQUE avoids such errors by restricting the use of focus to monotonic queries; errors which do arise are, at worst, sins of omission.

The problem of inappropriate application of focus may be ameliorated in several ways. The simplest is to provide feedback, to inform the user whenever a contextual, as opposed to a global, interpretation of his query has been chosen. PIQUE uses a simple natural language generation module for this purpose. For the example of section 1, the statement produced would be:

> By "Jones", I assume you mean the employee
> "Jones" with occupation = "programmer"

Feedback of this form does not solve the problem of inappropriate use, but warns the user of the possibility of errors.

Another method of avoiding error is to be "conservative" in the use of the focus mechanism – to avoid context-directed interpretation if there is doubt as to whether that is the user's intent. In PIQUE, this decision about appropriateness of the focus is made heuristically, based on two rules:

(1) If the query identifies a subset of an entity set which is already restricted by the focus space, do not use the contextual interpretation.

This prevents situations such as:

> Q1: Which employees work in the sales department?
> R1: Kegan, Desjardins, ...
> Q2: Which employees live in San Francisco?

The second query is presumably not intended to be interpreted in the context of the first. If the contextual interpretation had been intended, the user would probably have chosen a different formulation for Q2, such as "Which of them live in San Francisco."[*] The invocation of the focus mechanism here would be blocked by the fact that Q2 asks for the *name* (an identification field) in a subset of the set of employees.

(2) If the contextual interpretation of the query is already answered in the focus space, do not use the contextual interpretation.

Consider:

> Q1: List the name, salary, and occupation of employees who live in Palo Alto.

R1:	Name	Salary	Occupation
	Baker	20	clerk
	Allen	25	programmer
	Monro	30	programmer
	White	25	typist

> Q2: What are the salaries of the programmers?

In this case, interpreting "the programmers" as "the programmers who live in Palo Alto" leads to a reading which is already answered in Q1. The contextual reading of Q2 is *subsumed* by Q1; the test for subsumption can be performed directly on the DML expressions.

A third method for avoiding inappropriate use of focus is to operate in "failure-driven" mode – use focus only when the interpretation without context fails due to *reference failure*. An instance of reference failure appears in the example of section 1: the structure of the query Q2 indicates that "Jones' salary" is presumed to have a unique referent, but that is not the case for the global interpretation. The failure-driven method has not been implemented in PIQUE, because the DML does not have the capability of expressing assumptions of uniqueness, as occur in Q2 of the example. To encode such uniqueness, a *meta-query language*, like that used in (Kaplan, 1979) would have to be used. Also, the failure-driven approach does not handle plural noun phrases.

5. Other examples of use of this model

We discuss a number of other uses for a focus mechanism. These fall into two classes: (a) interpretation of potentially ambiguous user input; (b) other applications: updates, explanation, response generation, etc.

5. a) use of focus to aid in disambiguation

A natural language database system will often produce

[*] Pronominal reference presents different problems, and is typically addressed with different mechanisms; interpretation of pronouns is not implemented in PIQUE.

multiple interpretations of a single input. These all indicate the presence of some form of *ambiguity* in the user's input. In many cases, a preferred interpretation may be selected, using heuristics based on information contained in the database schema (e.g., Kaplan, 1979), or in the database itself (e.g., Harris, 1977). Interaction with the user (e.g., Codd, 1978) is another possible scheme for resolving ambiguity. The information provided by the focus model also allows use of heuristics that provide a preference ordering for the possible interpretations.

5.a.1) structural ambiguity – navigation

Natural language database systems must perform *navigation* between the concepts mentioned in a query, in order to reach an interpretation. If two concepts are multiply connected (i.e., the set of permitted connections specified in the database schema contains a cycle), each path between them will correspond to a different interpretation. The query might not provide enough information to determine the correct path. This problem might be called *pragmatic* or *structural* ambiguity, and can be addressed by a suitable context mechanism.

Consider a planning database, for an (American) city:

```
Residents
-------------------------------------------
| name | home-address | office-address | . . .
-------------------------------------------

Zip-Codes
----------------------
| address  | zip-code |
----------------------
```

and a simple dialogue:

Q1: What is the zip code of Forsythe's home address?
R1: 90120
Q2: What is Brown's zip code?

Q2, by itself, is ambiguous, since the connection between "Brown" and "zip code" within the database can be made in two ways, each involving a *join*:

```
{ y.zip-code : y∈zip-codes, x∈residents |
  x.name="Brown" ∧ y.address=x.home-address }

{ y.zip-code : y∈zip-codes, x∈residents |
  x.name="Brown" ∧ y.address=x.office-address }
```

Using the focus derived from the Q1, we see that the user is temporarily focusing on a particular sub-structure of the database, in which navigation between person and zip codes is performed via the home address. The structural similarity between the focus established by Q1 and one of the interpretations of Q2 can be recognized by observing that they have the same *connection graph*. Roughly, the connection graph is the representation of the connections between relations implied by the query (Ullman, 1980), and is sometimes formed automatically in course of query processing. Equivalence of connection graphs can be checked easily, allowing the preferred reading to be selected with the aid of the focus.

5.a.2) word-sense ambiguity

As in normal English, a word may have a number of different meanings in a database query program. Focus will often suggest the intended meaning. Consider a shipping database, which has information about classes of ships, including length, beam, draft, etc., and the following dialogue.

Q1: How long are tankers?
{where "tankers" are a class of ships}
R1: 240 ft.
Q2: Which is the biggest class?

"Biggest" has two distinct meanings in the lexicon, referring to the *length* of ships in a class, or the *number* of ships in the class. Given the dialogue above, the former reading seems to be preferred. That reading can be selected, using the fact that the attribute (property) mentioned in the query also appears in the focus.

Use of focus to aid in disambiguation, as in the above examples, is invoked only when global query interpretation has failed (produced multiple readings). Thus there is no danger of *inappropriate* use of the mechanism, although, of course, the heuristics themselves may be wrong.

5.b) other applications

In addition to aiding in query disambiguation, focus models may find application in a range of other database tasks.

5.b.1) database updates

The main goal of the PIQUE system is the interpretation of update requests. Processing updates expressed in natural language introduces problems beyond those encountered in processing natural language queries. These difficulties stem from the fact that the user will naturally phrase requests with respect to his conception of the domain, which may be a considerable simplification of the actual underlying database structure. Updates which are meaningful and unambiguous

from the user's standpoint may not translate into meaningful or unambiguous changes to the underlying database. Update requests may be *impossible* (cannot be performed in any way), *ambiguous* (can be performed in several ways), or *pathological* (can be performed only in ways which cause undesirable side effects). Natural language updates cannot be handled without some form of focus model; the model is necessary in order to generate the possible ways of performing the update, and to choose among them.

Consider a very simple database consisting of two relations:

Emps			Depts	
Name	Empno	Dept	Name	Mgr
Brown	103	Sales	Sales	Jones
Adams	222	Invntry	Invntry	Lasker
Larkin	145	Sales		

and the following dialogue:

Q1: List the employees, and the managers of their departments.

R1:

Name	Mgr
Brown	Jones
Adams	Lasker
Larkin	Jones

Q2: Change Brown's manager from Jones to Baker.

The update is a request to modify the information which was presented in R1. Since that relation is only *derived*, the change must be effected in the underlying database; the only reasonable way to do this is to replace Jones with Baker as manager of the Sales Department. (If Baker had been manager of another department, say Production, another possibility for performing the update would have been to move Brown to that department.)

Focus can be used in two ways here:

(1) Without consideration of Q1, Q2 may be rather meaningless;

(2) The update Q2 has the *side effect* of changing the manager of Larkin. PIQUE would inform the user of this with a message:

> Note that the employee Larkin has also had the manager attribute changed to "Baker".

Without a focus model, these side effects cannot be detected.

The processing of natural language updates with a focus model is analogous to the problem of performing updates through *views* of databases, which has been extensively studied (see, e.g., Keller, 1982; Dayal and Bernstein, 1978; Bancilhon and Spyratos, 1981). The difference lies in the short-term nature of the focus model, compared to the long-term character of views. For more details on the update process, see (Kaplan and Davidson, 1981).

5.b.2) response generation

The generation of descriptive expressions requires a model of the user's viewpoint, just as intepretation of such expressions does. The referring expression must be precise enough to enable the user to pick out the object(s) specified from all the other objects that he knows about.

Consider another update interaction with the database above:

Q1: List the employee numbers of employees in the sales department.

R1:

Name	Empno
Brown	103
Larkin	145

Q2: Change Brown's employee number to 222.

This time, there is exactly one way to perform the update: change Brown's employee number in the underlying database. If, however, there is a semantic constraint that employee numbers be *unique* (i.e., a functional dependency Empno --> Name), the change will be blocked because of the tuple (Adams 222 Inventory). The explanation given to the user by PIQUE would be:

> The EMPNO value of 222 has already been assigned to the employee Adams, whose department is "Inventory".

which informs the user of the existence of the "Adams" tuple, and indicates why that tuple was not seen previously. In this case, the model was necessary in order to understand which aspects of the tuple in question were *salient* to the user, allowing the system to present only those aspects.

7. Discussion and Conclusions

A method has been presented for modeling focus during natural language interaction with a database system, which enables the system to exhibit more appropriate behavior in certain situations than is otherwise possible.

Creation and maintenance of models of this sort are

inexpensive; the operations involved in retaining the focus, and deciding the applicability of a focus to a later query, are cheap. The test for applicability of focus will return quickly with negative results in most cases.

The view model does not require an explicit list of the information known by the user (as is the case in some other approaches), but rather operates with the *intensional* form of the user's view, resulting in a space efficiency. However, it may sometimes be desireable to retain the *extension* of the focus space, in the form of a "temporary" or "snapshot". In such cases, significant improvements in efficiency of query processing may result. In appropriate cases, the query may be evaluated against the temporary, rather than the full database, at much less cost; Finkelstein (1982a) considers this possibility in detail.

The approach described here does not require any additional information, beyond that which is already encoded in the database and schema. natural language capability (embodied in the grammar) need not be extended, since all operations are performed at the level of the DML. These points are requisites for *portability* of the natural language interface to a new domain or new database system.

Acknowledgements

This work was performed under ARPA contract # N00039-82-C-0250. The author would like to thank Beth Bottos for substantial contributions to this paper. Shel Finkelstein, Jerry Kaplan, and Arthur Keller made a number of helpful comments on earlier drafts.

References

Bancilhon, F., and N. Spyratos (1981). Update Semantics of Relational Views, *ACM Trans. on Database Systems*, 6,

Chen, P.P.S. (1976). The Entity-Relationship Model—Towards a Unified View of Data, *ACM Transactions on Database Systems*, 1

Codd, E.F., et al. (1978). Rendezvous Version 1: An Experimental English-Language Query Formulation System for Casual Users of Relational Databases, RJ2144 (29407) 1/26/78, IBM San Jose

Cohen, Philip R., and C. Raymond Perrault (1979). Elements of a Plan-based Theory of Speech Acts, *Cognitive Science*, 3, pp 177-212

Dayal, U., and P.A. Bernstein (1978). On the Updatability of Relational Views, *Proc. Fourth VLDB Conf*, 1978

Finkelstein, S.J. (1982a). Common Expression Analysis in Database Applications, to appear in *ACM SIGMOD International Conference on Management of Data*

Finkelstein, S. J. (1982b). Common expression analysis for optimization of database applications, Computer Science Department, Stanford University (to appear)

Grice, H.P. (1975). Logic and Conversation, in *Syntax and Semantics: Speech Acts*, vol 3, P. Cole and J.L. Morgan (eds), Academic Press, New York

Grosz, B.J. (1977). The Representation and Use of Focus in a System for Understanding Dialogues, *Proc 5th Int'l Joint Conf. Artificial Intelligence*, pp 67-76

Harris, Larry R. (1977). Natural Language Data Base Query: Using the data base itself as the definition of world knowledge and as an extension of the dictionary, TR 77-2, Math. Department, Dartmouth

Hendrix, G. (1977). Human Engineering for Applied Natural Language Processing, *Proc 5th Int'l Joint Conf. on Artificial Intelligence*, pp 183-191

Kaplan, S. Jerrold (1979) Cooperative Responses from a Portable Natural Language Data Base Query System, Ph.D. Dissertation, Computer and Information Science, University of Pennsylvania

Kaplan, S. Jerrold, and Jim Davidson (1981). Interpreting Natural language Database Updates, *Proc 19th Meeting of the Association for Computation Linguistics* pp 139-142

Keller, Arthur M. (1982). Updates to Relational Databases Through Views Involving Joins, to appear in *2nd Int'l Conf. on Databases: Improving Usability and Responsiveness*, Jerusalem, Israel

Rich, Elaine (1979). User Modeling via Stereotypes, *Cognitive Science*, 3, pp 329-354

Rowe, Neil C. (1982) Modeling Degrees of Item Interest for a General Database Query System, submitted to *International Journal of Man-Machine Studies*

Sidner, C.L. (1979). Towards A Computational Theory of Definite Anaphora Comprehension in English Discourse, MIT AILab TR-537

Stonebraker, Michael (1975). Implementation of Integrity Constraints and Views by Query Modification, *ACM SIGMOD Int'l Conf on Management of Data*, pp 65-78

Ullman, Jeffrey D. (1980). *Principles of Database Systems*, Computer Science Press, Rockville, Maryland

WHAT "ISA" IS AND ISN'T

Ronald J. Brachman

Fairchild Laboratory for
Artificial Intelligence Research

Palo Alto, CA 94304

1. Introduction

This is a treatise on knowledge representation - in particular, on the style of connecting up representational entities in semantic network and frame systems. A taxonomic hierarchy with some sort of "inheritance" link has been the mainstay of semantic nets since the work of Collins and Quillian (7, 8), and the "ISA link" (a.k.a. "IS-A", "IS", "SUPERC", "AKO", "SUBSET", etc., etc.) has been perhaps the most stable element of semantic nets as they have evolved over the years. But this stability is perhaps merely an illusion; as this paper* sets out to illustrate, there is often very little in common between ISA links in one system and another.

The idea of "ISA" is quite simple. Early in the history of semantic nets it was observed that much of representation of the world was concerned with the conceptual relations reflected in sentences like "John is a bachelor", and "a Dog is a domesticated carnivorous Mammal". That is, two of the predominant forms of statements to be handled by AI knowledge representation systems were the predicative one, expressing that an individual (e.g., John) is of a certain type (e.g., Bachelor), and the universally quantified conditional one, expressing that one type (e.g., Dog) is a "subtype" of another (e.g., Mammal). The easiest way to get such statements into a semantic net scheme was to have a link that directly represented the "is a" parts of the above sentences, and thus the ISA link was born.

It was quickly noted that the ISA connections formed a hierarchy (or in some cases a lattice) out of the types being connected. The hierarchical organization made it easy to distribute "properties" such that shared properties were stored at the place in the hierarchy that covered the maximal subset of nodes that shared them. This made the semantic net an efficient storage scheme, since shared properties were not replicated every place they held true; they were instead "inherited" by all nodes below the ones where they were stored. This, of course, is the notion of inheritance of properties that is always mentioned in the same breath as the ISA link.

Once the pattern of a network of ISA links with property inheritance was established, all kinds of new schemes developed that used the net as a basis for more elaborate kinds of statements, descriptions, etc. (see (5) for a survey). There also quickly arose a debate about whether the network structure was just so much obfuscation of the simple predicative and conditional statements that the ISA links were representing. It seemed that all such semantic nets provided was an indexing facility over formulae just as well (and perhaps better) expressed in the language of first order predicate logic (18, 12, 13, 15). The interesting thing about the debate was its consistent "apples vs. oranges" flavor - each time the logicians tried to pin down the intent of ISA, the net-workers would claim they were missing the point. The same was true of cross-net comparisons - one scheme was criticized on the basis of what the critic thought the ISA connections should mean, while the scheme was defended on the basis of what the author thought s/he meant.

If nothing else, the various debates over semantic nets have made it clear that there is not a single ISA link. It is also clear that little scientific progress can be made until we understand what the link could mean, since a coherent debate on the merits of logic vs. semantic nets cannot be had without some firm logical

*This paper is a brief summary of an invited talk of the same name to be presented at the Fourth National Conference of the Canadian Society for Computational Studies of Intelligence, Saskatoon, Saskatchewan, May 17-19, 1982. It is not intended to be a thorough treatment of the issues, but rather an outline of topics to be covered in the presentation.

claim about the import of ISA. (At the very least, it is hard to imagine trying to justify the advantage of a semantic net over logic without formally characterizing the expressive power of the former.)

So the question we set off to investigate here is this: just what is it that ISA links are intended to mean? In the course of this investigation we shall find some interesting and perhaps surprising things:

- the more or less "standard" use of ISA (the "default interpretation of ISA") has some serious problems. You cannot use a network based on it to represent complex concepts, and the notion of "cancellation" that follows from it can wreak havoc with your world knowledge.
- the tight association of inheritance and ISA serves only to confuse already confused matters further, and only by placing inheritance in its proper place (it is an implementation issue and not an expressive power one) can we get clear on what the claims about ISA really are.

Along the way, we will produce a rational reconstruction of the ISA relation, and make some constructive suggestions as to how the next generation of knowledge representation languages should be structured. We'll also have some fun with the "cancel link".

2. What ISA is

First we are going to attempt to catalogue the various semantic relations that ISA has been used to represent. This will most likely not be a complete catalogue, and it may even be unfair to certain network designers. But a somewhat careful look at the literature will reveal that this is such a murky area that perhaps we can be excused on these two counts.

2.1. An enumeration of ISA-intents

Before enumerating the ISA's, we need to quickly cover the kinds of things that ISA has been used to relate. This in itself complicates matters immensely - semantic net nodes (and frames, for that matter) have been variously thought of as representing sets, concepts, kinds, predicates, propositions, "prototypes", general terms, individual terms, and individuals (and probably many more things as well). One major split that we can make, despite this variety, is that between generic and individual interpretations of nodes. Roughly speaking, some semantic net nodes are thought to be descriptions that can apply to many individuals (think of "apply" here in the loosest sense possible), and some are thought to represent either descriptions applicable to a single individual, or such individuals themselves.

Generic nodes can be more or less specific than other generic nodes - this is what gives semantic nets their network structure - while individual nodes tend to be all at the same level of specificity.* Thus, all internal nodes in the network are generic, and the leaves are individual. So we can immediately divide the ISA relation into two major subtypes - one relating two generic nodes, and one relating an individual and a generic.** For example, if generic nodes are construed as sets and individual nodes as individuals (see, for example (14)), then we would expect to find an ISA for the subset relation, and one for the membership relation.

2.1.1. Generic/generic relations

When two generics are related by an ISA connection, the intent is usually that one is somehow related to, but less general than, the other. We have at least the following kinds of uses for

---

*Unfortunately, even this is controversial. Some authors distinguish between two kinds of individuals, roughly corresponding to "John" and "John as a child". The latter is sometimes called a "manifestation". Further, sometimes the individuals in semantic nets are considered to be descriptions and sometimes to be Russellian logically proper names (descriptionally vacuous).

**To the extent that an "IS" relation is considered a cousin of these ISA's (see, for example (1)), we also have a relation between two individual nodes to consider. We would consider at least the "is" of equality of individuals ("Cicero is Tully"), the "is" of attribution of an individual description to an individual ("Kareem is the tallest player"), and the "is" related to manifestation.

generic/generic relations:*

Subset/superset: when nodes are construed as sets, then the connection that gives the network its structure represents the subset relation. Some nets appear to represent relations like "a NUKE.SUB ISA SUBMARINE", but when these are sets, they really capture the proposition "all NUKE.SUBS are SUBMARINES". In other words, "for all x, if x is in the set NUKE.SUBS, then x is in the set SUBMARINES" is the meaning of ISA here.

Generalization/specialization: generalization seems to be expressible as a simple conditional: e.g., NUKE.SUB(x) $\supset$ SUBMARINE(x). This is the interpretation of the ISA link offered by Hayes (13), and is probably the standard semantic net connector. We should point out that something further needs to be said about the quantifier on the conditional - while Hayes speculates that a universal quantifier is what is meant, networks that interpret nodes as "prototypes" or somehow typical generics embed their conditionals in more unorthodox "defaults". We investigate this point further in Section 3, as it bears strongly on the expressive capability of the language using it.**

"AKO": "AKO" means "a kind of", and is intended to stand for the relation between "Camel" and "Mammal" in "the Camel is a kind of Mammal". To a very large extent, this is exactly the same as a generalization relation; however, somehow it feels wrong to have an AKO link from a concept that does not represent a kind (e.g., "a person who just happens to be walking to school right now" is certainly a person, but seems not to be a kind of person). Thus, we get the impression that one ought to distinguish between nodes standing for kinds and nodes standing for more arbitrary descriptions, thereby distinguishing between the generalization-style relation and the AKO-style relation.

Value restriction: another relation between generics is the kind intended in "the trunk of an elephant is a cylinder 1.3 meters long". The intent here is to say that a certain kind of entity (in this case, a "trunk") in some context must be of a certain type. This is related to "attribution" in (1).

Conceptual containment: in some cases, the intent of an ISA connection is not merely to state a generalization, but to express the fact that one description includes another. Instead of reading "a TRIANGLE ISA POLYGON" as a simple generalization (such that there are triangles and polygons and this happens to be the relation between them), we want to read it as "to be a Triangle is to be a Polygon with three sides". This is the "ISA" of lambda-abstraction, wherein one predicate is used in defining another. Note that this demands interpretation of nodes as structured descriptions, not simple predicates.

Set and its characteristic type: this isn't really an ISA relation; it's the one between the set of all Elephants and the concept of an Elephant. It associates the characteristic function of a set (e.g., a "prototype" in NETL (10)) with that set.

2.1.2. Generic/individual

The general intent of generic/individual connections is to state that an individual is describable by a general description.

Set membership: if the generic is construed as a set, then the relation is membership - "CLYDE ISA CAMEL" means "CLYDE is a member of (the set of) CAMELS".

Classification: this is the use of ISA that predicates a description of an individual. It usually involves a type predicate, like "DOG", or "BRICK".

Abstraction: this relation "goes the other way", in a sense. An abstraction is individual, like "the camel"; the abstraction relation is that between the singular description "the camel" and the (generic) predicate "camel(x)".

Conceptual containment: when the individual node is thought of as a structured individual description, the relation between it and a generic could be one of conceptual containment - the generic description could be used in the formation of the individual description. This is the case with the relation between, say, "the father of John" and "father" in "the father of John is a father". Note that the relation between the two occurrences of "father" in this statement is the classic type/token relation of the earliest semantic nets.

---

*Borgida (2) offers a similar treatment for the case where the objects related by ISA are procedures.

**Another related issue is the presence or absence of the "necessity" operator - see Section 2.2.4.

2.1.3. The "general purpose" ISA link

One approach to the plethora of ISA-relations that has surfaced from time to time is the "general purpose inheritance link". Since ISA has so many guises, its inventors argue, they are best served by making a "programmable" connection between nodes.* The user can turn off attributes that he doesn't want inherited by the more specific node, and turn on others that he does. Primitives like "PASS", "ADD", "EXCLUDE", and "SUBSTITUTE" (11) give the user extensive flexibility in making his ISA link do what he wants. The semantics of ISA relations constructed this way, however, cannot in general be predicted.

2.2. An analysis

There seem to be several different dimensions along which ISA links can vary. Each of these will be briefly described below:

- first, there is the type of conceptual entity that a node can embody (description, set, predicate) - this has a direct effect on the import of the ISA link (it governs the shape of what we'll call the "matrix" in Section 2.3);
- second, there is the basic syntactic function of the link - in particular, we contrast a sentence-forming intent with a description-forming one;
- third, for sentence-forming ISA's, we have a notion of the "quantifier" of the statement (e.g., honest-to-goodness universal vs. default);
- for these types of relations, we need also consider modality (necessity vs. possibility);
- finally, we must consider whether or not the link, by its very presence, makes an assertion.

---

*"...a knowledge representation system must represent arbitrary mappings between concepts." (11)

2.2.1. Effects of ontology

The first major influence on the ISA relation is the type of item that ISA is about. If ISA is a relation between two sets, then it is usually about their membership or cardinality. In the typical case, it is a relation between memberships, with an implicit statement about cardinality (the cardinality of the less generic node must be no greater than that of the one it is related to). The generic/individual version is typically the set membership relation (although see (14) for variations).

When the items to be related are predicates, then ISA typically has something to say about predications that follow from other predications, using the material conditional. The hierarchy derived from this style of ISA has an "if/then" flavor - if you are a person, then you are a mammal, etc. Note that this tends not to say what it is that makes you a person in the first place (see Section 3 for more on this). If the predicates related by ISA are all one-place, then the semantic net style link will suffice without elaboration. If, however, the predicates have arity greater than one, something must be done to account for the mapping of variables from antecedent to consequent. This has traditionally been done with slot names, but see (20, 4, 5) for a detailed discussion on the insufficiency of this mechanism.

When the ISA-related objects are intended to be descriptions, or "concepts", then the relation between them tends to be either about the structure of the descriptions themselves or about the classes of objects satisfying the descriptions. In the former case, an ISA like "a TRIANGLE ISA POLYGON" says that part of the description of any triangle is that it is a polygon. The same "kind of" relation holds if one of the descriptions is an individual description. Finally, when the ISA link is about the objects satisfying the ISA-related concepts (as opposed to being about the descriptions themselves), the relation is much like the subset relation.

It should be pointed out that the notion of an ISA relation carrying structure between structured descriptions is the point of most radical departure from standard predicate logic-based representation schemes. All of the other ISA sub-factors we have pointed out are easily expressed in standard quantificational languages (although see Section 2.3).

2.2.2. Sentence-forming vs. concept-forming

ISA links in most semantic nets are used to make statements about the world. When we say "an ELEPHANT-WITH-BLUE-EYES ISA ELEPHANT", we intend to make a statement about two classes: all things falling under ELEPHANT-WITH-BLUE-EYES also fall under ELEPHANT. If we want to be a little more explicit about the fact that these elephants have blue eyes, we might say "an ELEPHANT-WITH-BLUE-EYES ISA ELEPHANT, and has BLUE EYES". But let's not be misled by the hyphenated node name - this could just as well have read "a G0047 is an ELEPHANT and has BLUE EYES". This looks plainly like an assertion about some independently existing G0047 class.

But what about the class of elephants with blue eyes (no hyphens)? A different ISA-import is needed to create a description of that class. The standard sentence-forming ISA-import isn't enough, since we don't intend to imply that there is some independent class, elephant-with-blue-eyes, that is not merely the elephants with blue eyes. So there needs to be a distinct description-forming style of ISA to express the relation between the concept of an elephant with blue eyes, and the concept from which it is formed, elephant. This is what we have been calling "conceptual containment", above.

2.2.3. "A weak sense of 'every'"

As just mentioned, just about everyone uses the ISA relation to make a statement about two classes or a class and an individual. While the obvious quantifier to assume holding over such statements is the universal one (e.g., see (13)), this appears not to be in every semantic net designer's mind. For instance, consider,

> I am using a weak sense of the word "every" here: I mean that the property is true of every elephant for which it is not explicitly cancelled. (10)

Fahlman has an operator to take a "weak sense of 'every'" quantifier and make a standard one out of it - the "sacred" operator. So, for any sentence-forming ISA link, we need to know if it is a true universal, or merely a default.

2.2.4. Is this necessary?

While the distinction is almost never made in semantic net systems, there is another dimension along which ISA's can vary. Some statements, be they universal or default, could be otherwise - it is quite conceivable that, for example, banks in Massachusetts might have been open on Sundays; it just didn't turn out that way (i.e., "MASSACHUSETTS-BANK ISA COMMERCIAL-INSTITUTION-CLOSED-ON-SUNDAY" is contingently true). However, it is not possible that triangles could be anything but polygons. The latter is a necessary truth, the former a contingent one.

One property of the term-formation style of ISA is its implication of necessity for the concomitant relations (it is impossible for an elephant with blue eyes to fail to be an elephant).

2.2.5. To assert or not to assert

In many systems, the ISA link asserts a truth by its mere presence. Having the statement "CHRISTOPHER ISA SON-OF-J.R." in your semantic net means your system believes the horrible truth about Christopher. Without some other form of ISA relation, we are not free to contemplate a proposition without incidentally asserting it. Thus the distinction between asserted ISA's and simply structural ISA's adds another dimension to the ISA connection.

2.3. Why isn't this just logic?

Our analysis has left us with the following picture: there is a major split of kinds of things to say with ISA into

1. those that take one concept and form another out of it, and
2. those that make some sort of statement about the relation between two sets or the arguments to two predicates.

The ones that are used to make statements have four sub-components:

1. the "assertional force" of the statement - whether or not the statement represented by the ISA is to be considered believed.
2. the "modality" of the statement - whether the truth represented by ISA is necessarily true, or

is just contingently true (and could thus be contemplated to be otherwise).

3. the "quantifier" of the statement - whether the matrix is to be considered universally true, or just "true unless explicitly cancelled". In the latter case, the ISA is a default (see, for example, (19)).

4. the "matrix" - the content of the statement. As illustrated above, this generally has the structure of a set inclusion (or membership, in the case of a generic/individual ISA) or a material conditional (or predication, in the case of a generic/individual ISA) statement.

These four factors used to form the ISA relation look suspiciously like the pieces that make up more or less special cases of more or less standard logical statements (in, say, prenex normal form). Is the ISA link, then, accounted for completely by standard, off-the-shelf logical machinery? Why isn't this just logic?

Well, much of it is - but all of the factors combine to force us out of the realm of the standard, well-understood logics. We won't belabor the point here, since it is treated in depth in (15), but the modalities and defaults are enough to put us on shaky logical ground. When lambda abstraction, or something like it (the concept-forming kind of ISA), is added, then a semantic net account that is semantically well-specified is as valid a candidate for a logical account as something that looks more like predicate calculus.

In addition, there are other factors that make the concept-forming style of ISA and the resultant network-style languages look like real alternatives to standard predicate logic accounts.* For one, having structured terms that are interrelated provides a basis for a formal account of the terminology used to describe a domain. Predicates in standard predicate logic accounts are all atomic (see Section 3), and thus, the relations among the predicates themselves are not supported by the logic (remember above, where we mentioned that while it is easy to say "a person is a mammal", what being a person is in the first place is left unsaid). While this is not an issue of more expressive power (you can say what needs to be said in a primitive-predicate-based system), it is a matter of perspicuity, and perhaps even computational tractability.

Also, semantic net-style representation emphasizes certain compelling patterns in knowledge representation that do not emerge from predicate-logic based ones. For example, at least one interpretation of the concept/role paradigm (see 3.1 for the appropriate logical form) can be expressed easily in a standard logical language, but that pattern is just one among infinitely many. Network schemes have elevated the pattern to the level of a built-in form because of its widespread utility in representing knowledge. Another compelling pattern is the very distinction of "ISA" from "is" - semantic nets have acknowledged the prevalence of reasoning based on types from very early on, and have made a prominent distinction between the sense of "is" in "John is a man" and all other senses of "is" (e.g., "John is running scared", "John is extremely tall").* Some recent interest in using sorted logics for knowledge representation (e.g., in KLAUS (17)) indicates that this is another area where semantic net-based schemes can make a contribution.

So, in sum, it's not as if network schemes with their ISA-links are a non-contribution to the world of representation; it's just that we're usually pretty confusing about the nature of that contribution. Expressive power is not the crux of this knowledge representation issue; it's just one part of a multi-faceted job. We should try to be much clearer about the logical import of our networks, frames, or whatever, so that we can clearly see what is a real contribution, and what is just syntactic sugar. Section 4 provides an example of one aspect of the ISA link that has served more to confuse its import than to highlight its contribution.

3. What ISA shouldn't be

Now that we have elaborated a bit on the factors that make up the multi-faceted ISA link, there are two final observations to be made. The first, treated in this section, concerns the more or less

*These factors are discussed in more depth in (3).

*Notice that substituting an "is" link for ISA (1) undoes this distinction.

standard use of ISA links in modern semantic nets and frame systems. The second involves the way that "inheritance" fails to fit in with the rest of the semantics of the ISA relation, and is treated in the next section.

3.1. The default interpretation of ISA

As we mentioned above, Hayes (13) proposes the following as the meaning of a frame representing the concept C, with slot-relationships $R_1,\ldots,R_n$:

$$\forall x\ C(x) \supset R_1(x, f_1(x))$$
$$\& \forall x\ C(x) \supset R_2(x, f_2(x))$$
$$\& \ .\ .\ .$$

However, the standard use of ISA is as a default - "CLYDE ISA ELEPHANT" is a truth about Clyde until it is retracted ("cancelled"). This logical notation expresses the material conditional style of ISA, but not the default nature of it (remember that the standard ISA involves "a weak sense of 'every'").* The claim is made that without the ability to cancel properties, exceptions cannot be represented, and the world is such that exceptions are an important aspect of knowledge representation.

This style of representation (material conditional embedded in a default) strongly suggests that we think of the nodes in the net not as concepts, but simply as holding points for bundles of default properties. For example, a node like ELEPHANT should not be interpreted as representing the concept of a elephant, but instead as the place to find all of the properties of "typical" elephants. So, if we know "CLYDE ISA ELEPHANT", then we assume that all properties of the typical elephant hold of him.* At some point we may learn of some special feature of Clyde that distinguishes him from the prototypical elephant (say, for instance, that he has only three legs). We notate this by "cancelling" the normally inherited property (e.g., that typical elephants have four legs) and substituting the new one. An explicit cancelling mechanism allows accommodation of the fact that rarely do real elephants match their prototypes exactly.

Given that the properties of the prototype can be violated by instances of it, these properties are clearly non-definitional (which they would be if the "conceptual containment" style of ISA were used). This conclusion is reinforced by the "outward" nature of the slots of the frames: if Clyde is an elephant, then he has typical-elephant-properties - not the other way around (i.e., the connective in the above logical reformulation is the conditional, not the biconditional). Again, this seems well and good, since there are certainly no defining properties for elephanthood - the elephant is a "natural kind". And, you might add, so are most, if not all of the concepts that an AI system will have to deal with; leave abstract and defined concepts like RHOMBUS to the mathematicians, and leave the philosophers to argue about whether "bachelor" can be defined.

But this intuitively appealing and pervasive line is predicated on an interesting, though unargued and plausibly erroneous, assumption: as the elephant goes, so goes everykind else. The unwarranted belief that, with a few technical exceptions, every concept is natural kind-ish has had a significant consequence. There has been no felt need to provide a facility for expressing analytic or definitional connections. This perhaps raises no problems with the conceptual counterparts of lexical items like "elephant". But just as we can create the English phrase, "elephant that lives in Africa", we should expect to be able to create the node for the composite concept that it expresses. Two things are

---

*The default rules can be expressed as in Reiter's "Logic for Default Reasoning" (19), leaving the object language as is.

*Note that we could be tempted to interpret the node holding the elephant-properties as representing "the typical elephant". Fahlman (10) even goes so far as to label his nodes in that fashion. The major problem with this is that it is totally unclear what kind of a thing "the typical elephant is" - it certainly isn't any particular real elephant, like Clyde. And we don't want the ISA to mislead us into thinking that Clyde is the typical elephant.

certain - an elephant that lives in Africa can't fail to be an elephant, and it can't help but live in Africa! That is, the composite concept certainly stands in an analytic relationship to its "head" concept, even if that concept is associated with a natural kind.

The fact that defaults have been almost universally adopted at the expense of definitional or other analytic connections has left network representations in a funny state: the material conditional style of ISA works okay in one direction (Clyde will accrue the properties of elephants once you assert that he is one), and one can represent exceptions (three-legged elephants and the like). But one cannot represent even the simplest of conceptual composites. And it follows from the lack of composites that every single node (or frame or whatever) in the network is in fact semantically simple - in other words, a primitive. An AI system can certainly use such a network as a database repository for such classificatory facts as the user sees fit to tell it (e.g., CLYDE ISA TYPICAL-ELEPHANT), but it cannot draw any such conclusions itself. Without being told explicitly, the system cannot even tell that an elephant that lives in Africa is an elephant.

3.2. The myth of cancellation

The preponderance of default-style nodes in semantic nets has admitted cancellation of properties into the realm of representation. With it has unfortunately come a raft of technical problems (see (9), for example) - but even worse, the semantic consequences of cancellation have not been thought through. The intuitive feeling is that cancellation can be constrained to handle just the meaningful cases of exceptions; the truth is that cancellation admits the most bizarre structures with the most trivial amount of work.

For example, take this simple case. As we mentioned, if the nodes in a default-style semantic net are to be thought of as representing some kind of thing at all, they are best attributed representation of "typical" types of things. So the node labelled ELEPHANT would best be thought of as representing "the typical elephant". Let's say that we assert that "CLYDE ISA ELEPHANT", by which we really mean that Clyde is a typical elephant. Now let's say that poor Clyde has had a checkered past and lost one of his limbs in a street fight - we simply take advantage of the ability to cancel properties, and change Clyde's having four legs to his having three. Note that the ISA connection between Clyde and ELEPHANT is still there, all the while insisting that Clyde is a typical elephant. But our taking advantage of the typicality intent of ELEPHANT should have changed that - typical elephants have four legs and poor old Clyde has only three. Unfortunately, in default-style nets, there isn't any node to point to that would allow us to simply say "Clyde is an elephant, however many legs he has" - all of the nodes are just like ELEPHANT.

Anomalous behavior of all kinds can be generated from the standard ISA link and the concomitant association of typicality with the nodes it connects. For example, it's easy to imagine the ELEPHANT node having a connection that says "a TRUNK ISA CYLINDER with LENGTH 1.3 METERS" (see (16), ch. 6, p. 22, for example). Well, then, why isn't a BABY-GIRAFFE simply an ELEPHANT whose "1.3 METER CYLINDER" is its neck and not its trunk?

4. What ISA isn't

One important observation to be made about our analysis of the semantics of the ISA link is that "inheritance of properties" has played no part in our understanding. This is not without good reason - even though much has been made in the past of the significance of inheritance in semantic nets, no one has been able to show that it makes any difference in the expressive power of the system that advertises it. At best, any argument that inheritance was useful was made on pragmatic grounds: it saves storage space in an implementation.

Without denying the importance of implementation concerns, we submit that to the extent inheritance is a useful property, it is strictly an implementational one and bears no weight in any discussion of the expressive or communicative superiority of semantic nets. For one thing, any expression of properties at "the most general place" in a network-style system can duplicated easily in a logical one. You simply associate the property axioms with the most general predicate, and the standard conditionals do the rest. Further, inheritance is only one cut at the time/space tradeoff for storing properties in a semantic net; it may be tremendously easier in some cases to store all properties explicitly where they apply to

cut down search time.* Thus, although the ISA relation can be factored into sub-components, the useful ones for semantic purposes are assertional force, modality, etc., and not "pass this property" and "block this one". While these latter may be very useful tools for implementing a particular ISA methodology, they should not encroach on discussions of the adequacy of semantic net schemes for representing knowledge.

5. What ISA oughta be

What might be a viable prescription for future ISA-schemes? It is our feeling that the obvious one should be explored first. We advocate proceeding along the lines of our analysis of the import of ISA.

First, we should carefully distinguish between description- or term-forming operators and sentence-forming ones. There is a useful place for each, and a marriage of the traditional logical approach and the more recent "object-centered" terminological approach along these lines has not been explored. In (3), we explore some of the technical details of the marriage, and discuss in some depth the value of a completely definitional taxonomy over and above the "flat" logical axioms that it implies. We believe that structured predicates (or concepts) play an important role in expressing knowledge, and the vocabulary should be preserved in a representation, despite the fact that all statements using defined predicates might be reducible to a set of statements using only primitive ones. One way to do this would be to have a network-style representation scheme where the principal relation is the ISA of conceptual containment completely distinct from a network (or set of axioms) expressing the facts of the world. The latter set of statements (in the "assertional component") would use terminology from the former (the "terminological component").

The assertional component is where statements about the world are made. Thus, it needs to have the expressive and inferential power of at least standard predicate logic. This could be accomplished by using a standard quantificational language, or we could use a more network-like language. In the latter case, the backbone of the network would be the sentence-forming style of ISA.

This ISA could be broken down componentially into a "prefix" and a "matrix". The prefix would have three parts: the assertional force of the statement (whether or not it was to be believed), the modality (necessary, etc.), and the quantifier (universal, existential, "typical", etc.). The matrix itself would include conditionals much as in the "frame" equivalent above. What needs to be done is to specify what is implied by each kind of structure in the terminological component.

6. Conclusion

Semantic nets have prospered as a framework for knowledge representation, but all the while, their keystone construct - the ISA link - has wavered considerably in its interpretation. ISA has been used principally to form sentences that could be asserted - in particular, sentences with a default import. However, there are many other things that ISA has been used to mean, and comparison between networks and between networks and logic has been rendered all but impossible. The analysis presented here indicates that things might be a lot clearer if ISA were broken down into its semantic sub-components, and those used in turn to support representation (a similar kind of analysis should be done for the "PARTOF" or "HAS" link that semantic nets use to describe properties).

Finally, we should make it a habit to be careful about sprinkling talk of expressive power with implementation talk. Each has its proper place, but taken together, they tend to get confusing.

ACKNOWLEDGEMENT

Many thanks to Hector Levesque for helping me get much of this straightened out.

*In an implementation of KL-ONE (6), despite our purported "structured inheritance" framework, we ended up opting for "memo-izing" properties in order to cut down time searching up the network.

REFERENCES

1. Attardi, G., and Simi, M. Semantics of Inheritance and Attribution in the Description System Omega. A.I. Memo No. 642, M.I.T. Artificial Intelligence Laboratory, August, 1981.

2. Borgida, A. On the Definition of Specialization Hierarchies for Procedures. In Proc. IJCAI-81, Vancouver, B.C., 1981, 254-256.

3. Brachman, R. J., and Levesque, H. J. On the Marriage of Logic and Object-centered Knowledge Representations. Paper submitted to AAAI-82.

4. Brachman, R. J. A Structural Paradigm for Representing Knowledge. No. 3605, Bolt Beranek and Newman Inc., May, 1978.

5. Brachman, R. J. On the Epistemological Status of Semantic Networks. In Associative Networks: Representation and Use of Knowledge by Computers, Findler, N. V., (Ed.), New York, 1979, 3-50.

6. Brachman, R. J. An Introduction to KL-ONE. In Research in Natural Language Understanding, Annual Report (1 Sept. 78 - 31 Aug. 79), Brachman, R. J., et al., (Eds.), Cambridge, MA, 1980, 13-46.

7. Collins, Allan B., and Quillian M. R. Retrieval Time from Semantic Memory. J. of Verbal Learning and Verbal Behavior 8, 1969, 240-247.

8. Collins, Allan B., and Quillian M. R. Facilitating Retrieval from Semantic Memory: The Effect of Repeating Part of an Inference. In Acta Psychologica 33 Attention and Performance III, Sanders, A. F., (Ed.), Amsterdam, 1970, 304-314.

9. Fahlman, S. E., Touretzky, D. S., and van Roggen, W. Cancellation in a Parallel Semantic Network. Proc. IJCAI-81, Vancouver, B.C., 1981, 257-263.

10. Fahlman, S. E. NETL: A System for Representing and Using Real-World Knowledge. M.I.T. Press, Cambridge, MA, 1979.

11. Fox, M. S. On Inheritance in Knowledge Representation. Proc. IJCAI-79, Tokyo, Japan, 1979, 282-284.

12. Hayes, P. J. In Defence of Logic. Proc. IJCAI-77, Cambridge, MA, 1977, 559-565.

13. Hayes, P. J. The Logic of Frames. In Frame Conceptions and Text Understanding, Metzing, D., (Ed.), Berlin, 1979, 46-61.

14. Hendrix, G. G. Encoding Knowledge in Partitioned Networks. In Associative Networks: Representation and Use of Knowledge by Computers, Findler, N. V., (Ed.), New York, 1979, 51-92.

15. Israel, D. J., and Brachman, R. J. Distinctions and Confusions: A Catalogue Raisonne. Proc. IJCAI-81, Vancouver, B.C., 1981, 452-459.

16. Martin, W. A. Natural Language and the Computer Representation of Knowledge. Unpublished course notes, 6.863, M.I.T.

17. Moore, R. C., and Nilsson, N. J. Personal communication.

18. Reiter, R., and Criscuolo, G. Some Representational Issues in Default Reasoning. 80-7, Dept. of Computer Science, The University of British Columbia, August, 1980.

19. Reiter, R. A Logic for Default Reasoning. Artificial Intelligence 13, 1,2, 1980, 81-132.

20. Woods, W. A. What's in a Link: Foundations for Semantic Networks. In Representation and Understanding: Studies in Cognitive Science, Bobrow, D. G., and Collins, A., (Eds.), New York, 1975, 35-82.

DECOMPOSITION OF DOMAIN KNOWLEDGE INTO KNOWLEDGE SOURCES: THE MDX APPROACH

B. Chandrasekaran

Department of Computer and Information Science
The Ohio State University, Columbus, OH 43210 USA

Abstract

Our group's work in medical decision making has led us to formulate a framework for expert system design, in particular about how the domain knowledge may be decomposed into substructures. We propose that there exist different problem-solving types, i.e., uses of knowledge, and corresponding to each is a separate substructure specializing in that type of problem-solving. Each substructure is in turn further decomposed into a hierarchy of specialists which differ from each other not in the type of problem-solving, but in the conceptual content of their knowledge; e.g., one of them may specialize in "heart disease," while another may do so in "liver disease," though both of them are doing the same type of problem-solving. Thus ultimately all the knowledge in the system is distributed among problem-solvers which know how to use that knowledge. This is in contrast to the currently dominant expert system paradigm which proposes a common knowledge base accessed by knowledge-free problem-solvers of various kinds. In our framework there is no distinction between knowledge bases and problem-solvers : each knowledge source is a problem-solver. We have so far had occasion to deal with three generic problem-solving types in expert clinical reasoning : diagnosis (classification), data retrieval and organization, and reasoning about consequences of actions. In a novice, these expert structures are often incomplete, and other knowledge structures and learning processes are needed to construct and complete them.

Introduction

For the past two years our research group (consisting of the author, F. Gomez, S. Mittal and J. W. Smith, Jr.) has been investigating the issues of problem-solving as well as knowledge organization and representation in medical decision making. In parallel with this investigation we have also been building and extending a cluster of systems for various aspects of medical reasoning. The major system in this cluster is MDX, which is a diagnostic system, i.e., its role is to arrive at a classification of a given case into a node of a diagnostic hierarchy. The theoretical basis of this diagnostic problem-solving is laid out in some detail in Gomez and Chandrasekaran [1].

The MDX system, which is wholly diagnostic in its knowledge, communicates with two auxiliary systems, PATREC and RADEX. PATREC is a data base assistant in the sense it acquires the patient data, organizes them, and answers the queries of MDX concerning the patient data. In all these activities PATREC uses various types of inferential knowledge embedded in an underlying conceptual model of the domain of medical data. RADEX is a radiology consultant to MDX, and it suggests or confirms diagnostic possibilities by reasoning based on its knowledge of imaging procedures and relevant anatomy. See Mittal and Chandrasekaran [2] and Chandrasekaran et al [3] for further details about these subsystems.

Though in a sense RADEX and PATREC can both be viewed as "intelligent" data base specialists, RADEX has some additional features of interest due to the perceptual nature of some of its knowledge. However, for the purpose of this paper, it is not necessary to go into RADEX in much detail, and we can view PATREC as prototypical of this class of auxiliary systems.

Our aim in this paper is to outline a point of view about how a domain gets naturally decomposed into substructures each of which specializes in one type of problem-solving. Each of these substructures in turn further gets decomposed into small knowledge sources of the same problem-solving type, but specializing in different concepts in the domain. We shall see that this sort of decomposition results in more natural control and focus properties of the overall system. Identification of these substructure and how they communicate with one another is vital to the proper organization of the body of knowledge for problem-solving in that domain.

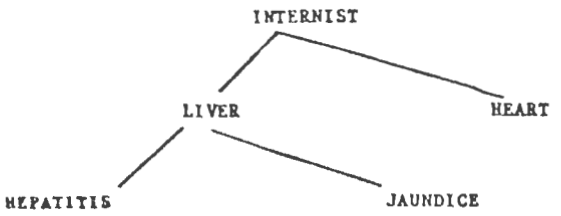
Our method in this paper will be to examine how knowledge is used in a few well-defined tasks: diagnosis, data storage and retrieval, and reasoning about consequences of actions. It should be emphasized that these tasks are not particular to the medical domain. Rather they are very fundamental generic tasks occurring in a wide variety of problem-solving situations. Thus these tasks are elements of a taxonomy of basic problem-solving types. When we are done with this examination, the general principles of knowledge decompositon will begin to take on some clarity.

One final point: we will use examples from both medical and non-medical domains. In particular there are many similarities between reasoning about diseases and therapies on one hand and trouble-shooting and synthesis of corrective actions in complex engineering systems on the other.

The Diagnostic Task

By the term "diagnostic task," we mean something very specific: the identification of a case description with a specific node in a pre-determined diagnostic hierarchy. For the purpose of current discussion let us assume that all the data that can be obtained are already there, i.e., the additional problem of launching exploratory procedures such as ordering new tests etc. does not exist. The following brief account is a summary of the more detailed account given in [1] of diagnostic problem-solving.

Let us imagine that corresponding to each node of the classification hierarchy alluded to earlier we identify a "concept." The total diagnostic knowledge is then distributed through the conceptual nodes of the hierarchy in a specific manner to be discussed shortly. The problem-solving for this task will be performed top down, i.e., the top-most concept will first get control of the case, then control will pass to an appropriate successor concept, and so on. In the medical example, a fragment of such a hierarchy might be:



More general classificatory concepts are higher in the structure, while more particular ones are lower in the hierarchy. It is as if INTERNIST first establishes that there is in fact a disease, then LIVER establishes that the case at hand is a liver disease, while say HEART etc. reject the case as being not in their domain. After this level, JAUNDICE may establish itself and so on.

Each of the concepts in the classification hierarchy has "how -to" knowledge in it in the form of a collection of diagnostic rules. These rules are of the form: <symptoms> ——> <concept in hierarchy>, e.g., "If high SGOT, add n units of evidence in favor of cholestasis." Because of the fact that when a concept rules itself out from relevance to a case, all its successors also get ruled out, large portions of the diagnostic knowledge structure never get exercised. On the other hand, when a concept is properly invoked, a small, highly relevant set of rules comes into play.

The problem-solving that goes on in such a structure is distributed. The problem-solving regime that is implicit in the structure can be characterized as an "establish-refine" type. That is, each concept first tries to establish or reject itself. If it succeeds in establishing itself, then the refinement process consists of seeing which of its successors can establish itself. Each concept has several clusters of rules: confirmatory rules, exclusionary rules, and perhaps some recommendation rules. The evidence for confirmation and exclusion can be suitably weighted and combined to arrive at a conclusion to establish, reject or suspend it. The last mentioned situation may arise if there is not sufficient data to make a decision. Recommendation rules are further optimization devices to reduce the work of the subconcepts. Further discussion of this type of rules is not necessary for our current purpose.

The concepts in the hierarchy are clearly not a static collection of knowledge. They are active in problem-solving. They also have knowledge only about establishing or rejecting the relevance of that conceptual entity. Thus they may be termed "specialists," in particular, "diagnostic specialists." The entire collection of specialists engages in distributed problem-solving.

The above account of diagnostic problem-solving is quite incomplete. We have not indicated how multiple diseases can be handled within the framework above, in particular when a patient has a disease secondary to another disease. Gomez has developed a theory of diagnostic problem-solving which enables the specialists in the diagnostic hierarchy to communicate the results of their analysis to each other by means of a blackboard [4], and how the problem-solving by different specialists can be coordinated. See [1] for details. Similarly, how the specialists combine the uncertainties of medical data and diagnostic knowldege to arrive at relatively robust conclusions about establishing or rejecting a concept is an important issue, for a discussion of which we refer the reader to [5].

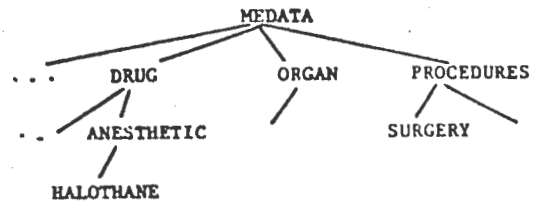
The points to notice here are the following. The control transfer from specialist to specialist is akin to the corresponding situation in the medical community. We shall have more to say about this later on. The most important point I'd like you to notice is that there is no "problem-solver" standing outside, using a knowledge base. The hierarchy of diagnostic specialists is the problem-solver as well as the knowledge-base, albeit of a limited type and scope. That is, the particular kind of problem-solving is embedded in each of the units in the knowledge structure.

Data Retrieval and Inference

Consider the following situation that might arise in diagnostic problem-solving that was discussed earlier. Suppose a rule in the liver specialist was: "If history of anesthetic exposure, consider hepatitis." This is a legitimate diagnostic rule in the sense described earlier, i.e., it relates a manifestation to a conceptual specialist. However suppose there is no mention of anesthetics in the patient record, but his history indicates recent major surgery. We would expect a competent physician to infer possible exposure to anesthetics in this case and proceed to consider hepatitis. Similarly, if a diagnostic rule has "abdominal surgery" as the datum needed to fire it, but the patient record mentions only biliary surgery, it does not take a deep knowledge of medicine to fire that diagnostic rule. In both these cases domain knowledge is needed, but the reasoning involved is not diagnostic reasoning in our specific technical sense. One can imagine an expert diagnostician turning, in the course of her diagnostic reasoning, to a nurse in charge of the patient record and asking if there was evidence of anesthetic exposure or of abdominal surgery, and the nurse answering affirmatively in both the instances without his being trained in diagnosis at all.

When we faced this problem in the design of MDX, we realized that it would be very inelegant to combine reasoning of this type with the diagnostic reasoning that we had isolated as a specific type of problem-solving activity. We were led to the creation of a separate subsystem for managing patient data, much like the nurse alluded to earlier. For all questions concerning manifestations MDX simply turned to this subsystem, which performed the relevant reasoning and returned the answer. We were surprised to discover that all the retrieval activities of this "data base assistant" could be captured in a uniform paradigm to be elaborated shortly. Mittal [6] describes this in detail as do the references [7], [8]. Similar to our discussion regarding the diagnostic task, we just touch upon the main issues here sufficient to make our main points regarding decomposition.

This data base — called PATREC — is organized as a hierarchy of medical data concepts. A fragment of this hierarchy is shown below.



At a representational level, there is nothing novel here: each medata concept is represented as a frame, and the inference rules that we will describe shortly are implemented as "demons" or "procedural attachments." However what will be worth noticing is the fact that all these rules will be of a certain uniform type. For the purpose of illustration, let us consider the SURGERY concept. SURGERY frame has LOCATION and PERFORMED? slots, among others. The "PERFORMED?" slot has the following rules:

1. If no surgery in the enclosing organ, surgery not done.
2. If surgery in a component, infer surgery in this organ.
3. If no surgery in any of the components, then infer no surgery in this organ.
4. If evidence of anesthetic, infer "possibly."

The DRUG frame has the following rules in the GIVEN? slot:

1. If any drug of this type given, then infer this drug also.
2. If the drug class was not given, rule out this particular drug.
3. If all drugs of this type were ruled out, then rule out the class too.

These rules need not be attached to the successors of DRUG, since they can inherit these rules — this is a fairly standard thing to do in frame-based systems. A successor may have further rules which are particular to it, e.g. the ANESTHETIC concept has the rule:

1. If major surgery, infer ANESTHETIC given, possibly.

Let us reemphasize that the interesting thing about the system is not that it uses frames with embedded production rules — by now it is a rare knowledge base system that doesn't — but that it is a collection of conceptual specialists tuned to a particular type of problem-solving. All the embedded inference rules have a common structure: derive the needed data value from data values relating to other concepts. The inferential knowledge that is encoded in the concepts is specific to the data retrieval task in a data base activity.

Let us consider some examples. Suppose the stored datum is that "Patient was given halothane." The HALOTHANE frame now has its GIVEN? slot filled with "Yes." Consider the following series of questions:

Q1. Given Anesthetic?
A : YES
(ANESTHETIC specialist inherits the rules from the DRUG frame. Rule 1 generates the question, among others, "Given Halothane?" "Yes" is propogated upwards.)

Q2. Any Surgery performed?
A : Possibly
(SURGERY specialist fails with rules 1, 2 and 3, rule 4 places query "Given

Anesthetic?" to ANESTHETIC specialist. "Yes" answer results in "Possibly" to Q2. This is an example of lateral inheritance.)

Similarly if the stored datum were "Patient had major surgery," and the query were, "Given Anesthetic?", rule 1 in ANESTHETIC would have given the answer "possibly."

Another, more complex example of data retrieval reasoning by PATREC is the following.

DATA: A liver-scan showed a filling defect in the left hepatic lobe. The liver was normal on physical exam.
Q : Liver Normal?
A : No
(On liver-scan data, following chain of inference took place: (a) filling-defect in lobe ——> lobe not normal; (b) If <comp-of> Liver not normal ——> liver not normal. On the other hand Physical examination produced "Normal" as answer. By using a general principle that when there are contending answers, non-default value should be chosen — the default for "Normal?" slot of LIVER is "Yes" — the answer "No" was generated.)

The main points relevant here are, as in the case of the diagnostic task: (1) There is no separation between a knowledge base and a problem-solver. Problem-solving is embedded in the knowledge structure. (2) All the conceptual specialists perform the same type of problem-solving, in this case, inheritance of data from other specialists. (3) Concepts with the same name, say LIVER, in the diagnostic structure and the data retrieval structure have different pieces of knowledge and do different things. This is akin to the fact that the LIVER concept of a diagnostician is bound to be different from that of the data base nurse. The concepts in this sense are "tuned" for different types of knowledge use.

What-Will-Happen-If or Consequence Finding

We said that among the many types of problem-solving that take place in a knowledge-rich domain is that of answering questions of the form "What will happen if X is done?" Examples are: "What will happen if valve A is closed in this power plant when the boiler is under high pressure?"; "What will happen if drug A is administered when both hepatitis and arthritis are known to be present?" Questions such as this can be surprisingly complex to answer since formally it involves tracing a path in a potentially large state space. Of course what makes possible in practice to trace this path is domain knowledge which contrains the possibilities in an efficient way.

The problem-solving involved and correspondingly the use of knowledge in this process are different from that of diagnosis. For one thing many of the pieces of knowledge for the two tasks are completely different. For example, consider answering the question in the automobile mechanic's domain: "What will happen if the engine gets hot?" Looking at all the diagnostic rules of the form, "hot engine -----> <malfunction>" will not be adequate, since <malfunction> in the above rules is the cause of the hot engine, while the consequence finding process looks for the effects of the hot engine. Formally, if we regard the underlying knowledge as a network connected by cause-effect links, where from each node multiple cause links as well as effect links emanate, we see that the search processes are different in the two instances of diagnosis and consequence-finding. The diagnostic concepts that typically help to provide focus and constrain search in the pursuit of correct causes will thus be different from the WWHI concepts that would be needed for the pursuit of correct effects.

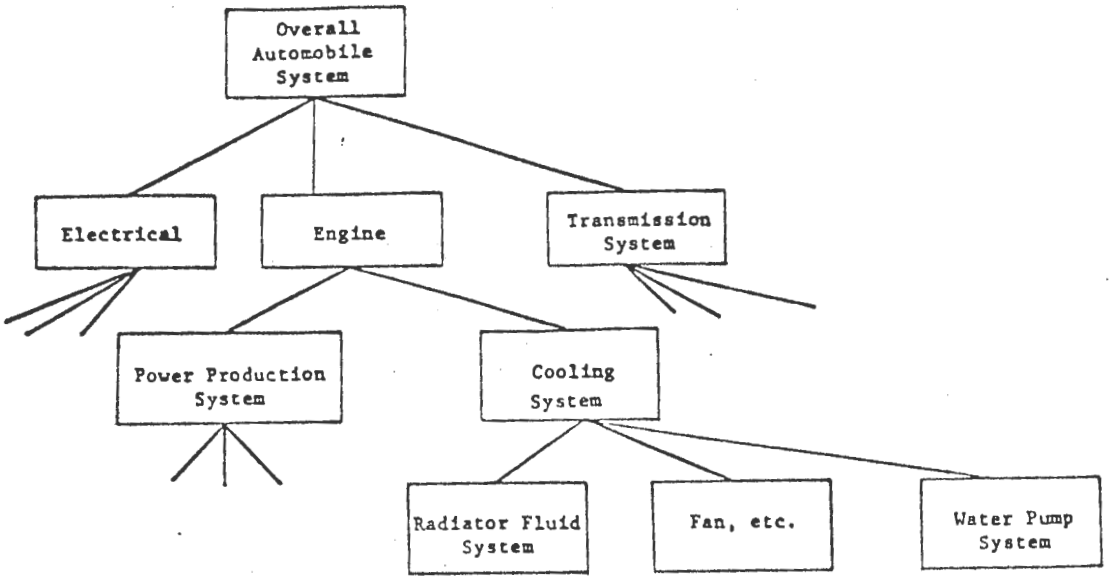
The embedded problem-solving is also correspondingly different. We propose that the appropriate language in which to express the consequence-finding rules is in terms of state-changes. To elaborate:

1. WWHI-condition is first understood as a state change in a subsystem.
2. Rules are available which have the form "<state change in subsystem> will result in <state change in subsystem>". Just as in the case of the diagnosis problem, there are thousands of rules in the case of any nontrivial domain. Again, following the diagnostic paradigm we have already set, we propose that these rules be associated with conceptual specialists. Thus typically all the state change rules whose left hand side deals with a subsystem will be aggregated in the specialist for that subsystem, and the right hand sides of those rules will refer to state changes of the immediately affected systems.

Again we propose that typically the specialists be organized hierarchically, so that a subsystem specialist, given a state change to it, determines, by knowledge-based reasoning, the state changes of the immediately larger system of which it is a part, and calls that specialist with the information determined by it. This process will be repeated until the state change(s) for the overall system, i.e., at the most general relevant level of abstraction, are determined. This form of organization of the rules should provide a great deal of focus to the reasoning process.

An Illustrative Example

Consider the question, in the domain of automobile mechnanics, "WWHI there is a leak in the radiator when the engine is running?" We shall first suggest the specialists are to be organized as follows:



The internal states that the radiator fluid subsystem might recognize may be partially listed as follows: {leaks/no leaks, rust build-up, total amount of water,... } similarly, the fan subsystem specialist might recognize states {bent/straight fan blades, loose/tight/disconnected fan belt,...}. The cooling system subsystem itself need not recognize states to this degree of detail; being a specialist at a somewhat higher level of abstraction it will recognize states such as {fluid flow rate, cooling-air flow rate...etc..}. Let us say that the radiator fluid specialist has, among other, the following rules. The rules are typically of the form: <internal state change> -----> <supersystem state change>

leak in the radiator -------> reduced fluid flow-rate

high rust in the pipes -------> reduced fluid flow-rate

no antifreeze in the water and very cold weather -------> zero fluid flow etc.

The cooling system specialist might have rules of the form

low fluid-flow rate and engine running -------> engine state hot

low air-flow rate and engine running -------> engine state hot

Again note that the internal state recognition is at the appropriate level of abstraction, and the conclusions refer to state changes of its parent system.

It should be fairly clear how such a system might be able to respond to the query about radiator leak. Again a blackboard for this task would make it possible to take into account subsystem interaction.

Unlike the structures for the diagnostic and data retrieval tasks, we have not yet implemented a system performing the WWHI-task. While we cannot speak with assurance about the adequacy of the proposed solution, we feel that it is of a piece with the other systems in pointing to the same set of morals: embedding still another type of problem-solving in a knowledge structure, which consists of cooperating specialists of the same problem-solving type.

Role of Common Sense and "Deep" Knowledge Structures

Recently Hart [9] has claimed that expert systems which use causal or other knowledge will perform better than systems which use "compiled" knowledge. Our foregoing discussion indicates that it need not be so. Suppose our diagnostic structure is missing the rule, "If high SGOT, add n units of evidence in favor of cholestasis." Then clearly if a knowledge structure involving knowledge of underlying biochemistry of liver function were available, and if the problem-solver knew how to use that knowledge, then it could get out of the diagnostic structure temporarily, access the biochemical/physiological knowledge structure, do some problem-solving, actually derive the rule relating SGOT and cholestasis, and re-enter the diagnostic structure to proceed to establish or reject cholestasis on the basis of this new-found

knowledge. However once the rule is derived it is not necessary to invoke the "deeper" knowledge structures. Similarly the role of common sense knowledge and learning structures is to create new pieces of knowledge in other structures, as productive thinking results in new understanding when new situations are faced and solved. However, we hold that, in principle, there exist "complete" problem-solving structures for diagnosis, consequence-finding etc. Given such idealized experts, it is not necessary to have the so-called deeper knowledge structures. However, real-life experts often may have to access the underlying structures in the absence of such "compiled" rules.

Knowledge-Use Taxonomy

There has been a growing realization in the field that the important issue in knowledge systems is to determine how knowledge is to be used. Our foregoing examination of the three tasks -- each of which is not some ad hoc need for medical reasoning, but is a generic task that arises in a number of domains -- leads us to propose the following theses.

(1) There is taxonomy of problem-solving regimes that are involved in expert problem-solving. We have identified three members of this taxonomy
- diagnostic (classificatory): establish-refine, top-down.
- consequence-finding: abstract state from low-level description to higher-level description, bottom-up.
- data retrieval: inheritance/inference of values from data values in other concepts.

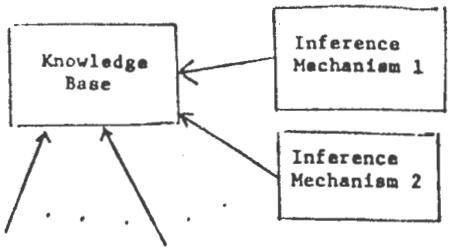
There are obviously more. Our research is oriented towards finding more elements of this taxonomy and determining their interrelationships.

(2) For each type of problem-solving there is a separate knowledge structure, with the associated p.s. regime embedded in it. Thus a domain of knowledge can be decomposed into a collection of structures, each of which specializes in a p.s. type. We can call this a horizontal decomposition of a domain.

(3) Each of the structures in (2) above can be further decomposed into a collection of specialists, all of whom are of the same p.s. type, but differ from each other in the conceptual content. We have indicated how this decomposition can be done for the three tasks considered. We term this decomposition a vertical decomposition.

A Paradigm Shift

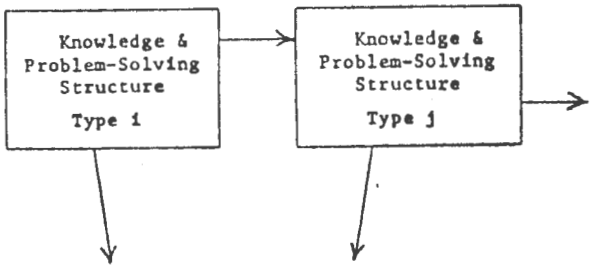
The prevalent approach to knowledge base systems is based on the following decomposition:



In this paradigm, knowledge representation is separated from its use. This approach has the attraction of generality and a certain kind of modularity.

The representational questions are dealt with in this approach in a manner to satisfy the criteria of expressiveness, or the so-called epistemological adequacy of McCarthy [10]. The efficiency responsibilities are put on the shoulders of the inference mechanisms; they have to have the so-called heuristic knowledge in order to use the knowledge efficiently for problem-solving. Our approach is based on a rather different decomposition of the same problem, as indicated in our discussion on horizontal decomposition in the previous section.

Pictorially, the viewpoint of knowledge based system that we advance can be given as follows:



Thus the overall knowledge system is viewed as a collection of specialists in inference types, who cooperatively solve a given problem. While in the figure we have indicated the communication among these specialists to be unconstrained, in fact, however, it may not be so. There may be reasons why only certain problem-solving specialists can talk to other problem-solving specialists. This is an open research problem in our approach.

Production Rule Methodology

In most of the preceding discussions the representation of knowledge has been in the form of rules. We feel that this is not accidental, but that rules represent a basic form of cognition, viz., "how-to" knowledge. This was

recognized early in AI by Newell and Simon [11] and they gave it the name production rules. Later, the Stanford Heuristic Programming Project extended this production rule methodology for a wide class of expert system design problems. We are thus in agreement with the use of rules as a basic knowledge representation formalism in expert systems.

There are two aspects in which our methodology differs from current work on rule-based expert systems. We have already alluded to the difference in the viewpoint which regards knowledge not as an independent structure to be used by different problem-solvers, but as embodiments of implicit problem-solving knowledge. Related to that is the idea that the central determinant of effective use of knowledge is how it is organized. Our approach begins to provide criteria for performing the organization of a complex body of knowledge. It is well-known that production rules need to be organized not simply for purposes of efficiency, but for focus and control in problem-solving (see [12] for a discussion of these issues). We are proposing two organizing constructs, which extend the production rule methodology to make it applicable to a larger class of problems. One construct is the problem-solving regime and the other is that of a conceptual specialist.

Related to these organizational notions is the other aspect of the difference between our approach and the current production rule methodologies. We do not use uniform problem-solving mechanisms (backward chaining, e.g.) across the whole domain. As indicated, the problem-solving method differs from knowledge structure to knowledge structure.

The Organization of the Medical Community

Evidence of Horizontal Decomposition

The medical community collectively is a good case study in the principles by which knowledge may be structured for cooperative, effective problem-solving. Corresponding to our notion of horizontal decomposition along the lines of problem-solving types, we can identify clinicians, educators, pathologists, radiologists, medical records specialists, etc. Clinicians combine the diagnostic and predictive knowledge structures, for practical reasons having to do with the close interaction between diagnosis and therapy. Medical record specialists, as their name indicates, serve to organize patient data and retrieve them effectively. Radiologists are not diagnosticians in the same sense as clinicians are: their problem-solving is to reason from imaging descriptions to confirm or reject diagnostic possibilities; they are largely perceptual specialists.

Evidence of Vertical Decomposition

Corresponding to our vertical decomposition, many of the above problem-solvers are organized into conceptual hierarchies. For instance, an internist is the top-level diagnostic specialist, who may call upon liver or heart specialists for further investigation of a problem. The top-down problem-solving for diagnosis is indicated by the fact that a sick person typically first goes to an internist, who may refer the patient on to more detailed specialists.

Evidence for Embedding Problem-Solving

If the medical community were organized according to the currently accepted paradigm in expert systems, i.e., a common knowledge base shared by different problem-solvers who themselves are without domain-knowledge, one would expect to have knowledge-specialists, who would be rather like encyclopaedias, and problem-solving specialists who would possess expert-algorithms for, say, diagnosis, without any medical knowledge about particular medical concepts. Thus whenever a patient came, the diagnostic specialist would consult the knowledge-base specialist and together they would arrive at a diagnostic conclusion.

However, that is not the way the community works. Instead we find that experienced medical specialists possess expertise which is not a raw knowledge-base, but which is highly effective in problem-solving. On the other hand, a medical student without clinical experience is more like a pure knowledge-base. As he or she becomes more experienced in various types of problem-solving, the unstructured knowledge base slowly begins to shape and structure itself, so that pieces of knowledge are tuned for ready use.

Acknowledgements

I owe a great debt of intellectual gratitude to Fernando Gomez, who was responsible for many of the central theoretical ideas behind MDX. Sanjay Mittal has been a dedicated co-worker, responsible for much of the implementation as well as for many of the ideas incorporated in PATREC and RADEX. Jack Smith, M. D., is, in addition to being the medical component of our group, also a skilled computer scientist, which contributed greatly to a smooth AI/medical interface.

The preparation of this paper was supported by NSF grant MCS-8103480. The computing resources of Rutgers University Laboratory for Computer Science Research were used extensively in system development, and this was made possible by the Biotechnology Resources Division of NIH, under grant RR-00643.

References

[1] F. Gomez and B. Chandrasekaran, "Knowledge organization and distribution for medical diagnosis," IEEE Trans. Systems, Man Cybernetics, Vol. SMC-11, No. 1, January 1981, pp. 34-42.

[2] S. Mittal and B. Chandrasekaran, "Conceptual representation of patient data bases," J. Medical Systems, 1981.

[3] B. Chandrasekaran, S. Mittal and J. Smith, "RADEX - towards a computer-based radiology consultant," in Pattern Recognition in Practice, E. S. Gelsema and L. N. Kanal (Ed.), North-Holland, 1980.

[4] L. D. Erman and V. R. Lesser, "A multi-level organization for problem-solving using many diverse cooperating sources of knowledge," in Proc. 4th Int. Joint Conf. Artificial Intelligence, 1975, pp. 483-490.

[5] B. Chandrasekaran, S. Mittal, and J. W. Smith, M.D., "Reasoning with uncertain knowledge: the MDX approach," Proc. 1st Ann. Joint Conf. of the American Medical Informatics Association, May, 1982.

[6] S. Mittal, "Design of a distributed medical diagnosis and database system," Ph.D. Thesis, Dept. of Computer and Information Science, The Ohio State University, 1980.

[7] S. Mittal and B. Chandrasekaran, "Software design of knowledge-directed database systems," Proc. 1st Conf. Foundations of Software Technology and Theoretical Computer Science, Tatu Institute of Fundamental Research, Bombay, India, Dec., 1981.

[8] S. Mittal and B. Chandrasekaran, "Some issues in the design of knowledge-directed databases," Working Paper, AI Group, The Department of Computer and Information Science, The Ohio State University, 1981.

[9] P. E. Hart, "Direction for AI in the eightiea," ACM SIGART Newsletter, #79, January 1982, pp. 11-16.

[10] J. McCarthy and P. J. Hayes, "Some philosophical problems from the standpoint of artificial intelligence," in Machine Intelligence, Vol. 4, New York: Elsevier, 1969.

[11] A. Newell and H. A. Simon, Human Problem Solving, Englewood Cliffs, NJ: Prentice-Hall, 1972.

[12] D. B. Lenat and G. Harris, "Designing a rule system that searches for scientific discoveries," in Pattern-Directed Inference Systems, D. A. Waterman and F. Hayes-Roth, Eds., New York: Academic Press, 1978.

Three Flavors of Parallelism

Scott E. Fahlman
Department of Computer Science
Carnegie-Mellon University
Pittsburgh, Pennsylvania 15213

Abstract

This paper explores the relative costs and powers of three different kinds of parallel computing architecture that have been proposed for use in AI. Instead of parcelling a problem out to a small, fixed number of processors, all of these systems employ a much higher degree of parallelism: they provide enough processing elements that we can assign a separate processor to every assertion in a large knowledge base, to every pixel in an image, or to every word in a speech system's lexicon. But, while these three kinds of system share this general orientation toward the massive use of parallelism, they differ markedly in the complexity of their processing elements and interconnections. They also differ in the kinds of problems that they can attack in parallel, without resorting to serial processing techniques. In order of increasing complexity and power, these categories are *marker-passing* systems, *value-passing* systems, and *message-passing* systems.

1. Introduction

For several years, my students and I have been exploring the use of specialized, highly parallel computing architectures for a variety of AI tasks, especially the task of representing and accessing large amounts of real-world knowledge. Most of this work has centered around the NETL system [3], a design that is at once very powerful and very limited. On the one hand, NETL uses a separate hardware processing element for every entity and every simple assertion in its knowledge base. Thus, for a large class of problems involving searches and simple deductions in this knowledge base, the available processing power grows at the same rate as the problem. The time required for such operations is constant or nearly constant, regardless of the size of the knowledge base. In a very large knowledge base, such a system can perform *some* tasks thousands or millions of times faster than a traditional serial processor built from comparable technology. On the other hand, the NETL processing elements are so simple and their abilities are so limited that some tasks cannot be attacked in parallel at all; these tasks must be dealt with serially, just as in a traditional machine. In such cases, all of the parallel processing hardware buys you nothing.

The problem is that we have not had a clean way of characterizing which tasks a NETL-like marker-passing system can handle in parallel and which tasks must, unavoidably, be handled serially. We have never been quite sure, when NETL is unable to handle some problem in parallel, whether the problem is inherent in the limitations of the architecture or is merely a result of our particular choice of notations and algorithms. Other architectures, more powerful and much more costly, have been proposed from time to time, but without a good theory of the powers and limitations of the various parallel architectures it has been impossible to see clearly what the more complex hardware buys you.

In wrestling with these problems over the last few years, we have learned a great deal. We still do not have anything as elegant as the theory of Turing computability for these parallel systems, but we can precisely characterize some of the things that a marker-passing system can do, and we have some examples of tasks that definitely *cannot* be done in parallel by such a system. We have also discovered that some of the tasks that are impossible for a marker-passing system to perform in parallel can be handled by other, more complex parallel architectures. This paper will explore three classes of parallel systems that seem to form a hierarchy of increasing complexity and increasing power.

NETL is an example of the simplest of the three classes, which I call "marker-passing" systems. These pass only a few types of single-bit messages, called "markers", among the processing elements and can perform only simple Boolean operations on them. More complex are the "value-passing" systems, which can pass around numbers or continuous values and do simple arithmetic operations on them as they flow through the network. Most complex are the "message-passing" systems, which can pass around messages of arbitrary complexity and can perform arbitrary computations on them within each element.

2. The NETL Architecture

The NETL architecture can be taken as the stereotypical marker-passing system, though many variations on the basic theme are possible. Let us begin, then, with a brief description of NETL. Readers already familiar with NETL can skip this section.

NETL is a descendant of the semantic network memory first proposed by Quillian [11]. Each primitive concept in the memory (elephant, Clyde-the-elephant, gray, truth, last-Thursday, etc.) is represented by a *node*, which in NETL is a very simple hardware device. (See figure 1.) Each node communicates with the central control computer, a serial computer of the familiar kind, via a shared party-line bus. Each node has a unique serial number by which it can identify itself to the central controller and which can be used as an address when the controller wants to send a command to just one node. Inside each node there are a few (perhaps 16 or so) independent bits of memory that can be set and cleared by various commands on the control bus. These are called the *marker bits*: if bit 3 is on, then the node is said to be marked with marker 3. There are also a few write-once type bits that record some additional information about the node: whether it is representing an individual or a class, etc.

Each simple assertion in NETL, for example "Clyde is an elephant", is represented by a *link*. A link is also a simple hardware element, and it also communicates with the central control computer over a shared bus. A link has a few write-once type bits, plus a number of wires (typically four), each of which can be connected to any node in the network. To store an assertion, the control

computer must locate an unused link element and permanently connect its wires to the appropriate nodes. To store "Clyde is an elephant", for example, an unused link has its wires connected to the *Clyde* node, the *elephant* node, and the *is-a* node. Normally, two of the wires, called "A" and "B", are tied to the nodes that the assertion is about (*Clyde* and *elephant*, in this case). A third wire, called "parent" is tied to a node representing the type of relationship ("is-a"). The fourth wire, "context", is tied to a node representing the context in which the assertion is stated to be true; if the statement is universally true, it is tied to the *everywhere-and-always* node.

These connections must be private-line connections, since information (in the form of marker bits) will be flowing over many of them at once -- that is one source of the system's parallelism. The connections must also be permanent; if a connection is broken, the system forgets the assertion that the link is trying to represent. For most purposes, it is sufficient to think of the connections as physical wires that are soldered into place when a new fact is learned, but in a practical NETL machine these connections would actually be set up in a switching network, similar in design to a telephone switching system but designed so that all the subscribers can be talking at once.

And so, a working NETL system consists of a serial control computer connected to a huge number of node and link elements[1] wired up to represent the entities and relationships of some body of knowledge. The control computer can order an individual node, addressed by its serial number, to set or clear a certain marker, or it can address a command to whole classes of nodes: "All nodes with both marker 2 and marker 3 on, set marker 4." This command intersects two marked sets: if all the nodes representing members of one set are marked with marker 2 and all the members of another set are marked with marker 3, this command puts marker 4 on the members of the intersection-set in a single cycle. A serial machine would have to consider every member of one of the sets individually to perform the same task, so the time required would be proportional to the size of the smaller set.

This operation of intersecting stored sets, sometimes very large ones, is a common and important one in AI, particularly in recognition and knowledge-accessing tasks. Very often, for example, we must in effect ask the knowledge base whether there is any stored description in memory that exhibits a particular set of features. We might want to know, for example, if there is any disease characterized by nausea, headaches, a rash, and no fever. This is basically an intersection task: we have sets of diseases that are associated with nausea, with fever, with no fever, and so on, and we must find the intersection set. Serial AI systems either settle for a very small knowledge base and scan it linearly, or they pick certain features as the most important and index the memory using these. The problem with the latter approach is that, in noisy real-world situations, we cannot be sure that the indexing features will be among those that were observed; if they are not, we must either fall back on linear search or give up altogether. So the ability to intersect stored sets in parallel is very useful in AI tasks.

The control computer can also address a command to a whole class of links: "Every *is-a* link whose A-wire goes to a node with marker 3 on, tickle the node on your B wire. Every node that feels itself being tickled, and that does not already have marker 3 on, set marker 3 now and indicate that something changed by tugging on the response line of the shared control bus." This command propagates marker 3 one level up the hierarchy of *is-a* relationships. from all the nodes that currently have marker 3 to their immediate superiors. We could, for example, mark the *Clyde* node with marker 3 and then, by repeatedly giving the above command, mark all of Clyde's superiors in the *is-a* hierarchy with marker 3 as well. When nobody tugs on the response line, the propagation is complete, and all of Clyde's superiors have been marked.

Operations of this sort give us a way to perform the transitive closure operation over networks of *is-a* relationships in time proportional to the length of the longest chain of links that must be followed. On a serial machine, this operation would take time proportional to the total number of nodes in the tree. Because of branching, this might be very much larger than the height of the tree. We can also perform fast transitive closures over other transitive relations: *part-of*, *before*, *bigger-than*, and so on. Note, however, that the operation can only proceed in parallel if the relationships that form the tree are explicitly present in the network; computing and adding new links is a serial operation in NETL.

This ability to do transitive closures quickly is very useful in practical AI systems, especially in knowledge-base systems in which the properties of the typical member of some class are stored only with the node representing that class and are inherited by the subclasses and their members. To find the color of Clyde, we mark the Clyde node with marker 1, then propagate this marker to every type-node above Clyde in the *is-a* hierarchy; we have now found all of the classes from which Clyde is supposed to inherit properties. Then, in a single cycle, we can look for any *color-of* links emanating from one of these marked nodes. In this case, we would find the *color-of* link between *elephant* and *gray*. This operation is very quick because the branches of the *is-a* tree are never very long, though the tree may be very bushy and a node may have many superiors. Without a parallel transitive closure mechanism, we must either store information redundantly redundantly at each node or take care to limit the amount of branching in the *is-a* hierarchy.

In addition to intersection and transitive closure, the NETL architecture makes it possible for a marker on one set of nodes to gate the flow of other markers through the network. For example, the controller could direct marker 1 to flow up *is-a* links, but only to cross those *is-a* links whose context wire is connected to a node with marker 2 set, and not to enter any node that has marker 3 set.

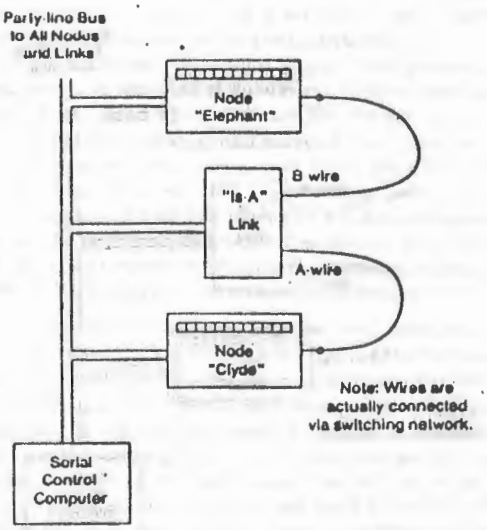


FIGURE 1: NETL SYSTEM HARDWARE

[1] In some versions of NETL, there is only one type of element that can be used either as a node or a link.

This gating ability is also very useful in knowledge-representation systems. Some assertions in a knowledge base are universally true, but many are only true at certain times, certain places, or in some alternative reality such as a novel. By tying each link to a context node and by marking the set of context nodes that are to be active at any given time, we can perform searches and deductions using only those assertions whose context nodes are active; links whose context nodes lack the "active" marker simply play dead. The context nodes are connected into a complex, tangled hierarchy similar to the hierarchy of *is-a* realtions, and many of them are normally active at once, each with many associated assertions. For example, if it is January 20 in Canada, it is also winter, and in the winter context, it is cold, certain animals are white, others are hibernating, and so on. The parallel gating mechanism gives us a way of dealing with complex sets of overlapping contexts, each with its own set of assertions, at essentially no extra cost in access time.

3. Marker-Passing Systems

A key observation about NETL-like marker-passing systems is that they are about as cheap and simple as any massively parallel system can be. It is true that NETL requires one element for every entity and every simple assertion in the knowledge base, but these elements are very simple. A typical system might have storage for 16 volatile marker bits and for 16 write-once type bits per element. All that a NETL element really contains is these 32 bits of state, some logic to perform simple boolean operations on them, and a control-bus interface that can be shared among all the elements on an IC chip. If the number of private-line connections required were not a problem, then with today's technology one could easily fit 1000 or more of these elements onto a single chip.

However, the elements are not the whole story, or even the most important part of it. Most of the long-term knowledge in a NETL system is stored in the pattern of interconnections between the link-wires and the nodes. It is the switching network that contains most of the state of the system, and not the elements themselves. This switching network implements many-to-one connections: each link wire goes to only one node terminal, but a node terminal may be connected to many link wires. The signals are simple binary levels, and they are OR'ed together at the node terminal. If a particular marker is being propagated from link wires to nodes during a given cycle, a node does not have to record how many instances of that marker have arrived or where they came from; it simply records that marker N was received. Of course, when a marker is moving from a node to the attached link wires, each link wire gets a copy.

This OR'ing together of the markers arriving at a node is the defining feature of marker-passing systems. It means that the switching network can be quite simple, while still guaranteeing that there is no contention. If a million instances of marker 3 arrive at a node, they can all be accepted at once, in one cycle, and the arrival of marker 3 can be recorded in one bit of memory within the node. During this cycle, marker 3 is the only message being passed, so no confusion can result.

In fact, it would be impractical to tie a million wires directly to a single node terminal. A real marker-passing machine would use a multi-level switching network, and would allow connection paths bound for the same node to merge at any level in the network. The result is a tree of connections, with OR'ing and amplification occurring at each level of the network. See [4] for a sketch of how such a system might be built for a million-element NETL Machine.

We have seen that marker-passing systems of this sort, despite their simplicity, offer us some definite advantages over serial machines for certain AI tasks. We have gradually come to understand that they suffer from some fundamental limitations as well, and we are beginning to understand what these limitations mean in AI contexts.

The key limitation is in the nature of the markers themselves: only a small, fixed number of markers are available, and it is impossible for an individual marker to record where it came from or what path it followed through an arbitrarily complex network. This means that certain apparently simple operations cannot be done in parallel. Suppose, for example, we want to mark the set of all sons who are hated by their own fathers[2]. We can easily mark all the males with marker 1, their fathers with marker 2, and everyone hated by a father with marker 3. Every node with both markers 1 and 3 is a son who is hated by *some* father, but is he hated by *his own* father? The system has no way to tell, except by serially inspecting each of these nodes and seeing if the hater and the father are one and the same. Some additional parallel operations can be performed to narrow down the list of suspects -- for example, we can eliminate any son, hated by someone, whose father hates no one -- but we cannot get the desired answer without some serial processing of individual nodes.

What is needed in this situation is a more complex form of marker. Suppose that each node could paint the markers it sends out with its own unique color of paint (a colorful version of its serial number). Then each father could send a painted marker 1 to each of his sons and a painted marker 2 to everyone that he hates. The son-nodes could then compare the color of the marker 1 it receives with the colors on all of the marker 2's that arrive. (Some of the sons might be hated by lots of fathers.) If there is a match, the son is hated by his own father. This comparison happens to be one-to-many, but if the problem were to identify all of the people who are hated by any of their siblings, the match would be many-to-many; the question would be whether any marker 1 has the same color as any marker 2 received by the same node.

Note that this use of painted markers is inconsistent with the definition of a marker-passing system that we gave above: since the color of each of the incoming nodes must be preserved, we cannot just OR incoming nodes together and receive them as a single message. Instead, we must accept each incoming marker individually and store it away for later comparison, at least in the many-to-many case. Since there is no *a priori* bound on the number of links that might be connected to a node, there is no limit to the number of messages that might show up at once, and no limit to the amount of storage required to hold on to them. If the incoming messages are all to be accepted at once, we need an unbounded number of input ports on each node. More likely, we use a single input port and queue up incoming messages; this means that we must be prepared to tolerate unbounded delay due to contention at the most popular nodes. In short, by allowing painted markers with an unlimited number of colors in the system, we have crossed the boundary from simple marker-passing parallelism to the much more complex message-passing parallelism, which we will discuss later.

So, if we are to remain within the cheap, contention-free marker-passing discipline, we must live with the fact that some operations cannot be performed without resorting to some amount of serial, node-by-node processing. There are several ways to characterize this class of problems. In the language of relational data bases (see, for example, [2]), the operations we cannot handle are called *joins*. In the language of the lambda calculus, we can only have as many distinct lambda-bindings in effect at one time as we have

[2] This example was first suggested to me by Brian Smith.

markers to represent them. The problem above is that we want to look at all of the possible bindings of *father* at once, but we must keep the various bindings of *father* distinct and not treat them as interchangeable members of a set.

Some other problems we have encountered in NETL seem to arise from this same fundamental limitation. The copy-confusion and binding-ambiguity problems discussed in section 3.7 of Fahlman [3] could easily be solved in a system with painted markers, but they require some amount of careful serial processing if they are to be handled correctly in NETL. Similarly, most of the problems we have had with cancellation (see [5]) would go away if we could write arbitrary messages on markers as they pass through certain nodes in the network: when a marker passes the tail of a cancel-link, the link would add a spot of its own color of paint to the marker; when a marker passes the head of a cancel-link, it would be stopped and inspected, and if it has that link's paint on it, it would be eliminated. In effect, each node must be able to carry with it a history of where it has been, and there is no *a priori* limit on how much information such a history might have to include. Since, in a pure marker-passing system, we cannot paint the markers, we must either do some serial processing when a cancel-link is encountered during a propagation, or we must do some serial analysis whenever the network changes and store the result as redundant auxiliary links in the network.

4. Message-Passing Systems

Let us turn now to the most complex and powerful of the three families, the *message-passing* parallel systems. As stated earlier, these systems allow arbitrary messages to be passed around the network, can store any number of messages within the element (in practice, of course, there is some limit, but it can be taken to be very large), and can perform arbitrary Turing-machine computations on the stored messages. The signed-marker algorithms discussed in the previous section are among the simplest uses of message passing parallelism; more complex are the "society of agents" models of Minsky [10], and Hewitt's "actors" [12]. Clearly, if machines in this class must resort to serial processing, it is due to some fundamental constraint within the problem -- there is no more powerful class of parallel machine to which we can appeal, short of an oracle.

One problem with machines of this class is that they are hard to program and to control. If the elements are communicating via arbitrary messages, they must all speak the same language and obey the same conventions. The more complex the messages, the more difficult it is to set this language up and maintain it as the system grows and changes. If some of the elements are, in effect, able to control the operation of others, then careful attention to the control mechanisms is required to prevent chaos on the one hand and various sorts of deadlocks on the other. These problems may not be insurmountable, but they are difficult. By contrast, all of the complexity in NETL lies in the serial programs in the control computer; the elements are too simple to get into any mischief.

Of course, to simplify the communication and control problems, we may choose to use a message-passing machine in very limited ways. The painted-marker scheme is an example of such a limited style, really no harder to program and analyze than NETL itself.

But even if we leave aside the programming problems, a message-passing system remains very much more expensive to implement than a marker-passing system. Each individual element must be some sort of Turing-equivalent microprocessor, and there must be a large amount of storage for the saved messages, perhaps a few thousand words per node. Instead of getting a thousand of these on a chip, we would be lucky to get one. Contention in the network is inevitable, and buffering for stacked-up messages must be provided. If the complex messages are to be transmitted bit-serially, keeping down the cost of the network hardware, this makes the potential for delay even worse. So, while these systems can do in parallel some of the things that NETL must do serially, the "parallel" operations themselves may introduce unbounded delays.

One ingenious solution to this dilemma is currently being explored by Hillis [7]. Hillis allows only a small, fixed number of connections to be made to each element, limiting the number of messages that can arrive there at once and therefore also limiting the amount of storage needed and the amount of delay due to contention. A node, in the NETL sense, is represented not by a single physical element, but by many, connected in a balanced tree structure. When the need arises to add more connections to a node, more hardware elements are recruited and are added to the tree of elements that already comprises the node. Thus, as the number of potential incoming messages increases, the number of input ports and the amount of internal storage in the node increases proportionately. Unfortunately, since the node is no longer atomic, there is now the need for internal message traffic and the possibility for contention within the data paths internal to the node. This would be especially bad if the problem involved a many-to-many match of the stored messages, as in the "hated by any sibling" problem described above. Still, for some tasks this approach holds promise, and it will be interesting to see how far Hillis and his colleagues can take it.

So, the message-passing systems escape from some of the problems that force a marker-passing system into serial-processing mode, but with a fundamental and very substantial increase in cost. An interesting question, then, is whether this added power is worth the cost. How important is it that we be able to perform arbitrary join operations very fast? Can people do this? How close can we come to handling copy confusion and cancellation properly with marker-passing, and how much trouble will be caused if a few cases slip through? Can we at least determine, in parallel, that we have encountered a case where a correct answer will require serial processing? On the other hand, if we do need message-passing, how bad will the contention and memory problems be in real problem domains? Though unbounded in principle, these things might be quite tractable for many domains of interest. All of these are interesting questions, and at the moment I can't answer any of them. At least, given this structuring of the problem, we can begin to look for answers.

5. Value-Passing Systems

The value-passing systems are intermediate in complexity and power between the marker-passing systems and the message-passing systems. Instead of passing around discrete marker-bits, the value-passing systems can pass around numbers or continuous quantities with some limited amount of precision -- eight bits, for example. Such values carry more information than a simple marker, but they are not kept separate as true messages must be. If several values arrive at a node at once, they are added together and only the sum is received and recorded. Alternatively, the maximum or minimum of all arriving values might be saved. Other combining functions might also be used, such as multiplication or averaging, but these are more difficult to implement; for now, we will allow only *sum*, *min*, and *max*.

If a multi-level switching network is used, the connections to a single node will form a tree, with values coming together at each of

layers. Since the combining operations are associative and commutative, the same combining rule may be used at each place where a new value joins the tree. Each element must provide storage for a few of these values, and must be able to do a few simple arithmetic operations on them: comparison, negation, scaling by a constant factor, and so on. All of these operations are under the control of a serial computer, just as in NETL.

It should be clear that a value-passing machine of this sort can easily simulate a marker-passing machine, so its capabilities must be a superset of those possessed by NETL. It should also be clear that, since messages are combined rather than being handled separately, this a value-passing machine cannot do a parallel join any better than a marker-passing machine can. However, this same property of combining messages makes it possible for a value-passing machine to get by with bounded memory requirements and it guarantees that no contention can occur in the network. In fact, a value-passing machine is more complex than a marker-passing machine fo the same size, but only by a constant factor, perhaps a factor of 4. By contrast, a message-passing machine could easily be hundreds of times more complex, depending on how much "unbounded" memory is present in each node. Of course, all of these numbers are very rough; the machine designer has considerable freedom to trade off complexity against speed.

This particular combination of value-passing with massively parallel hardware and with some sort of sequential control is relatively new. Some of the same elements were present in the old Perceptron models and in such abstractions as the fuzzy logic of Zadeh [13], but in a different mixture and a different context. The current wave of research in this area has been influenced by some recent findings in neuroscience and by the work of David Marr, and it tends to emphasize problems in vision and motor control. See Hinton [8], Feldman [6], and Ballard [1] for examples. One interesting development is the use of what Hinton calls a "mosaic" representation: where some analog value, such as the angle of a line in the visual field, is to be represented, its range is quantized into a number of smaller ranges, each represented by a node. The activation on each of these nodes is a measure of how strongly the system believes that that particular value is the right one.

Value-passing systems share many of the fundamental limitations of marker-passing systems, in that they cannot perform parallel joins and cannot perform the painted-marker operations described above. However, they do have some advantages over marker-passing systems in other areas. Often the few discrete markers of NETL are too coarse for the task at hand. Consider, for example, the kind of recognition problem described earlier. If we have only a few discrete features to deal with, we can assign one marker to each feature and look for descriptions that have collected all of these markers. But if we do not find such a description, we must consider descriptions that have collected only four out of five markers, or three out of five -- in other words, we must resort to a sort of scoring system. At least, we need some arithmetic mechanism in each node that can count up the number of markers present. If we have many features, a few of which will be spurious, and if the features are of different strengths in helping to select or reject certain hypotheses, a true value-passing system is needed. It allows each observed feature to vote for some collection of hypotheses, and to vote for each with a different strength. If one description gathers many more votes than its competitors, that description wins.

This sort of system meshes very well with NETL-type marker passing: at the more abstract levels, marker-passing deduction can be used to select a set of possible descriptions. The nodes in this set are marked, and the mark is cashed in for some amount of initial activation. The observed features then cast their votes and a winner is determined. The winner gets a new mark, and this can be used in subsequent high-level processing.

Even more interesting is the possible use of value-passing in what might be called Gestalt recognition, in which the parts of an image cannot be recognized out of context, and the context can only be determined by use of the parts. For example, the words "base", "diamond", "pitcher", "strike", and "ball" are ambiguous individually, but taken together they clearly select a baseball context; this, in turn, selects a particular meaning for each of the words. This is a simple, discrete example that can be handled adequately by a marker-passing machine, but in many tasks the features are more numerous and, individually, less powerful. In such cases, we again need a mechanism by which features can vote for the hypotheses of their choice, and the hypotheses can reinforce certain interpretations of the features. This is a complex feedback process, and due care must be exercised to prevent instability, but I believe that it may offer some real hope for solving real-world recognition problems that otherwise would be intractable. Work in this area is just beginning, however, and it is too soon to have much evidence to back up this hunch.

There are some other places in NETL where the ability to pass around values would help us out of awkward problems. In understanding English text, we often favor one meaning for a word over another on the basis of the general context we are in and the "semantic distance" between each meaning and the other items that have been mentioned recently. To pick a very simple example, if we have rooks and pawns around, the word "king" tends to be interpreted as a chess piece; if we have crowns and thrones around, we get the other meaning. This is just a predisposition; other evidence can override this selection. A voting system works much better for this than the discrete marker-passing of NETL. With a voting system, we can take into account how many items we have seen that would reinforce each meaning, how long ago these items were mentioned, and how close the relationship is between each meaning and the other items in the context.

Attempts to implement something like the Harpy speech-understanding system [9] in NETL have also demonstrated the need for continuous values. In such systems, we create a few new hypotheses at each step of the analysis, and we must keep scores for several hypotheses at once, since the top-rated hypothesis may well be knocked out by new evidence. In general, a marker-passing style seems to work well at the higher conceptual levels of the system, except when we are doing the sort of free-association described above -- Clyde is either an elephant or he is not. Continuous values seem to become more important as we get closer to the inputs and outputs of the system. If we build everything out of value-passing hardware, we can easily mix marker-passing and value-passing styles of operation, using each where it seems most appropriate.

6. Summary and Conclusions

A few points should by now be clear:

- The marker-passing parallelism style of NETL helps us to avoid many problems that have plagued serial approaches to AI.
- Some of the fundamental limitations of marker-passing create very awkward problems for NETL. Some kinds of tasks must be done in serial if they are to be done properly.

- Message-passing eliminates many of these problems, but because it must handle messages separately, it is much more expensive to implement and suffers from contention in the network.

- Value-passing parallelism is only slightly more expensive than marker-passing, and is contention-free. It offers some advantages over marker-passing, especially in low-level I/O domains, but it suffers from many of the same limitations.

It is too early to draw any sweeping conclusions from all this. My own guess is that full message-passing will prove to be unnecessary in creating a human-like intelligence and that some mixture of marker-passing and value-passing will suffice. My own immediate plan is to work with value-passing systems to learn mroe about what they can do and how they can be combined with marker-passing systems like NETL. I may also try to implement a value-passing machine in hardware, if simulation of these systems proves to be inadequate. Once these systems are well-understood, we will be able to see more clearly whether they are sufficient for the task. The similarity between some of the value-passing models and some models coming out of neuroscience is encouraging, though certainly not conclusive. Some good people are working on message-passing systems, so that approach is covered as well.

In any event, by sketching out this hierarchy of parallel systems, I hope to have contributed some useful structure to the ongoing exploration of massively parallel systems for AI. New parallel models and interesting hybrids will certainly arise, and the hierarchy of types described here may help us to classify and understand these new systems.

Acknowledgements

I would like to thank Dave Touretzky, Walter van Roggen, Geoff Hinton, Jerry Feldman, Dana Ballard, Danny Hillis, and Gerald Sussman for helping me to recognize some of the limitations of NETL and for suggesting some of the alternatives explored in this paper.

This research is supported by the Defense Advanced Research Projects Agency, Department of Defense, ARPA Order 3597, monitored by the Air Force Avionics Laboratory under contract F33615 78 C-1551. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the U.S. Government.

References

1. Ballard, D. H. Parameter Networks: Towards a Theory of Low-Level Vision. Proceedings, IJCAI-7, 1981.
2. Codd, E. F. "A Relational Model of Data for Large Shared Data Bases." *CACM 13* (June 1970).
3. Fahlman, S. E.. *NETL: A System for Representing and Using Real-World Knowledge*. MIT Press, Cambridge, Mass., 1979.
4. Fahlman, S. E. Design Sketch for a Million-Element NETL Machine. Proceedings, AAAI Conference, 1980.
5. Fahlman, S. E., Touretzky, D. S., and van Roggen, W. Cancellation in a Parallel Semantic Netowrk. Proceedings, IJCAI-7, 1981.
6. Feldman, J. A. A Connectionist Model of Visual Memory. In *Parallel Models of Associative Memory*, G. Hinton & J. A. Anderson, Eds., Lawrence Erlbaum Associates, Hillsdale, N. J., 1981.
7. Hillis, W. D. The Connection Machine. Tech. Rept. 646, MIT Artificial Intelligence Laboratory, 1981.
8. Hinton, G. E. Shape Representation in Parallel Systems. Proceedings, IJCAI-7, 1981.
9. Lowerre, B. T. The HARPY Speech-Recognition System. Carnegie-Mellon University, Department of Computer Science, 1976.
10. Minsky, M. L. Plain Talk About Neurodevelopmental Epistemology. Proceedings, Fifth International Joint Conference on Artificial Intelligence, IJCAI-5, Cambridge, Mass., 1977.
11. Quillian, M. R. Semantic Memory. In *Semantic Information Processing*, Minsky, M., Eds., MIT Press, Cambridge, Mass., 1968.
12. Yonezawa, A. & Hewitt, C. Modeling Distributed Systems. Proceedings, Fifth International Joint Conference on Artificial Intelligence, IJCAI-5, Cambridge, Mass., 1977.
13. Zadeh, L. A. A Theory of Approximate Reasoning. Tech. Rept. UCB/ERL M77/58, Electronics Research Laboratory, U. of Cal., Berkeley, 1977.

LOOKING AT LEARNING

ROGER C. SCHANK

YALE UNIVERSITY
COMPUTER SCIENCE DEPT.
NEW HAVEN, CT 06520

Although learning is one of the most important areas for research in the field of artificial intelligence, very little has been achieved in the way of a general theory of human learning. This is due, we believe, to the need for learning to take place within the framework of a general understanding system. We believe that research at Yale in the field of building such an understanding system has prepared us to work on a model of learning.

We propose four basic criteria that a model of learning can be judged by:

1. FORM - The structure of what is to be learned

2. CONTEXT - The conext provided by the understanding system for the learning of new items

3. DEVELOPMENTAL - Proposing a model that is consistant with known facts about the stages of human learning

4. EVOLUTIONARY - Proposing a motivation for learning in the context of understanding which will allow the system to continue to learn and evolve as its structures grow.

Our model is based on two ideas: Memory Organization Packets, or MOPs, which are our proposal for memory structures, and the concept of failure driven modification of these memory structures. Briefly, MOPs provide expectations about situations the understander is faced with, and the failure of these expectations is used to index similar episodes and to indicate modifications which must be made to the MOP structures.

We will discuss how our model addresses our four criteria. In addition, we will review briefly the work on learning to date in AI, and discuss how it addresses - or fails to address - these issues.

ASPECTS OF COMPUTATIONAL VISION

Robert J. Woodham

University of British Columbia

Abstract

Computational vision is the study of computational systems that interpret images. Representations required at different levels of visual processing are discussed. Some specific knowledge sources that constrain the descriptions produced at each level are presented. These include laws of physical optics and generic knowledge of possible configurations in the world. The transformation of a description of image features into a description of scene features is the crucial step in any computer vision system. The image irradiance equation, based on the reflectance map and the gradient space, provides an appropriate theoretical tool to understand and exploit this transformation. Examples are drawn from applications to industrial automation and remote sensing.

Introduction

Computational vision is the study of computational systems that interpret images. That is, it is the study of systems that produce symbolic descriptions of a world from images of that world. Computational vision can be distinguished from other disciplines that deal directly or indirectly with images along two principal dimensions: the problem addressed and the approach taken.

Computational vision is concerned fundamentally with images of three-dimensional worlds. A scene domain consists of objects and surfaces defined in three spatial dimensions. An imaging device projects rays of light onto a plane. The image domain consists of a spatially varying brightness function (image irradiance) defined over a bounded planar region. The problem of computational vision is to reconstruct a three-dimensional representation of the scene from its two-dimensional projection onto the image plane. More succinctly, computational vision produces symbolic descriptions of surfaces from measurements of image irradiance.

The approach taken is a computational one. That is, any vision system is regarded as a general information processor performing computations on and manipulating internal symbolic descriptions of visual information. Within this framework, key questions become: What representations are required at various levels of the computation? What information is being made explicit at each level of the computation? How are the various representations transformed from one level to the next? What additional constraints are required to move from one level to the next? Are these constraints generally valid?

Computational vision has developed its own paradigm for research. One aspect is to choose a stylized fragment of reality to study in depth. In early computational vision research, the fragment of reality was a world of plane-faced polyhedra, uniformly painted with matte white paint and assembled into structures on a flat table covered with matte black felt. The task of making sense of this blocksworld had a precise functional definition. The goal was to understand images well enough so that a copy of the scene in view could be automatically assembled using a robot arm and a warehouse of spare parts. This copy demonstration was the focus of research lasting for about ten years (1965-1975) [1-8].

Substantial progress has been made since 1975 both to define the levels of representation required in computational vision and to describe specific modules of vision working at each level. This is largely the work of David Marr and his colleagues [9] but has also benefited from the work of others [10, 11].

Four levels of representation emerge. First, there is the representation of locations in an image where intensity is changing abruptly. (Marr calls this the primal sketch). Second, explicit surface properties are assigned to the locations described in the primal sketch. (Marr calls this the raw 2 1/2-D sketch). Explicit surface properties include: reflectance changes; illumination changes (shadows, highlights, etc.); discontinuities in depth; discontinuities in surface orientation. Third, the explicit surface properties determined at specific image locations are interpolated so that the surface properties are defined everywhere in the image. (Marr calls this the full 2 1/2-D sketch. This level of representation has also been called an intrinsic image [11]. When the surface property in question is surface orientation, it has also been called a needle map [12].) These three levels of representation are all defined in a viewer-centred coordinate system. That is, they all define spatially varying functions over the plane forced by the imaging geometry. The final and fourth level produces full 3-D models of the objects in view defined in an object-centred coordinate system. This last representation is independent of viewer position.

Specific modules that have been studied in detail include: binocular stereo [13, 14]; shape from shading [10, 15]; photometric stereo [16]; shape from motion [17]; shape from texture [18]; shape from contour [19-21]; optical flow [22]. Modules which have so far eluded analysis include the human-like perception of lightness and colour.

Many lessons have been learned. Perhaps the most important is that writing computer programs to interpret images in a domain requires a careful teasing out of the relationship between objects in the world and their pictorial traces. This relationship defines the semantics of imaging for the particular visual world in question.

Computer vision systems do not have to be universal in order to be useful. A common sleight-of-hand occurs in making the transition between image features and scene features. Isolating where this transition occurs is the key to understanding when a particular system will work and when it will fail.

Typical application areas include: industrial automation (visual inspection, manipulation and locomotion); remote sensing (interpretation of aerial and satellite imagery); biomedicine (microscope images, radiographic images, tomographic reconstructions).

This paper accompanies an invited talk with the same title. The talk highlights the relationship between image features and scene features using examples from industrial automation and remote sensing. The purpose of this paper is to provide a summary of the issues discussed and an appropriate list of references.

Interpreting Image Features as Properties of the Underlying Scene

There is no simple correspondence between image irradiance, the quantity measured in an image, and properties of the underlying scene. Image irradiance results from the interaction of several factors, some of which are properties of the objects in view and some of which are not. The effects of those which are, surface shape and surface material, must be separated from each other and from the artifacts of those which are not, illumination, shadows, viewing direction and path phenomena.

Several examples serve to illustrate the difficulties. One example shows an egg carton with several compartments containing eggs, others empty. The sense of which compartments contain eggs is reversed when the same image is shown again upside down. This corresponds to a fundamental ambiguity in distinguishing indentations from protrusions. It is known that humans prefer an interpretation which corresponds to the illumination coming from above.

A second example is more subtle. An image is presented which corresponds to a right circular cone made of a certain material and illuminated from a particular direction. It is demonstrated that this image also corresponds to a right cone with elliptical cross section made of a slightly more reflective material and illuminated from a slightly different direction [15]. Trade-offs emerge between surface shape and surface material that cannot be resolved in a single view. Applications of computational vision often assume uniform surface material and known illumination so that changes in image irradiance can be reliably related to surface shape.

Further examples come from applications to remote sensing. Remote sensing often assumes a uniform surface and known illumination so that changes in image irradiance can be reliably related to surface material (i.e., ground cover). Computer-based remote sensing has been most successful for wheat and other crop inventories in the prairies of North America where, indeed, the ground is flat and changes in image irradiance are due principally to changes in ground cover. It is also not surprising that these same techniques applied to forest inventory in British Columbia do very poorly since changes in image irradiance are dominated by changes in surface shape (i.e., local topography) [23].

Methods for assigning surface properties to image features in a single view embody assumptions about surface shape, surface material and illumination conditions. To deal explicitly with these factors, it is necessary to understand how images are formed.

Developing the Image Irradiance Equation

Four factors interact to determine image irradiance. They are: imaging geometry, incident illumination, surface photometry and surface topography. The imaging geometry determines the projection of three-dimensional scene coordinates onto two-dimensional image coordinates. Without illumination, there can be no image. Incident illumination is characterized by the spatial and spectral distribution and state of polarization of radiant energy falling on the scene. Surface photometry determines how light reflects off a surface. It is determined by the optical constants of the material and by its surface microstructure. (Surface microstructure is surface detail which is too fine to be resolved in the image but which nevertheless alters the way light is reflected). Surface topography is surface detail which is within the resolution limits of the image sensor. It characterizes the gross object shape relative to the viewer.

Four images of sphere-like objects illustrate: an image of a full moon in which there is no change in image irradiance as a function of surface topography; an image of a shiny coloured billiard ball showing a component due to specular reflection; an image of a flat white ball showing characteristic Lambertian reflectance; an image of a pollen grain obtained with a scanning electron microscope (SEM) showing a non-intuitive yet easy to interpret pattern of shading.

An image irradiance equation can be developed to relate the geometry and radiometry of image formation [10, 15]. The reflectance map, originated by Horn, allows image irradiance to be written as a function of surface orientation in a viewer-centred coordinate system. The reflectance map

uses the gradient space, popularized by Huffman [4] and Mackworth [6] to represent surface orientation. The reflectance map provides a uniform representation for specifying the surface reflectance of a given material for a particular light source, object surface and viewer geometry.

In computational vision, the image irradiance equation is used to help analyze what is seen. Unfortunately, the reflectance map itself is not invertible since surface orientation has two degrees of freedom and image irradiance provides only one measurement. In order to determine the underlying scene, additional information must be provided.

Recent work has explored how constraints on surface curvature can simplify the analysis. Surface orientation can be determined locally for planar surfaces forming trihedral corners [10] and for developable surfaces and generalized cones [15]. These results help to delineate shape information that can be determined from geometric measurements at object boundaries and shape information that can be determined from intensity measurements over sections of smoothly curved surface.

The image irradiance equation also helps to analyze what isn't seen. Recent work on binocular stereo matches zero-crossing segments from a left and right view of a scene [13]. (Zero-crossing segments are the basic assertions of Marr's level one representation, the primal sketch). Specific surface properties are attributed and matched to produce a depth map at the level of the raw 2 1/2-D sketch. These depth values must be interpolated to produce the full 2 1/2-D sketch. The image irradiance equation constrains the surface interpolation in regions where there is an absence of zero-crossing segments.

In remote sensing, the image irradiance equation has been used to predict variations in image irradiance due to a known topography. This facilitates image rectification [24] and helps to delineate changes in ground cover from topographic affects in areas of rugged relief [23].

Photometric Stereo

Another way to provide additional constraint is to obtain multiple images. A novel application of the image irradiance equation is the technique called photometric stereo [16]. Binocular stereo determines range by relating two images of an object viewed from different directions. If the correspondence between picture elements is known then distance to the object can be calculated by triangulation. Unfortunately, it is difficult to determine this correspondence. The idea of photometric stereo is to vary the incident illumination between successive images, while holding the viewing direction constant. It is shown that this provides sufficient information to determine surface orientation at each image point. Since the imaging geometry is not changed, the correspondence between image points is fixed. The technique is photometric because it uses the radiance values recorded at a single image location, in successive views, rather than the relative positions of displaced features.

Photometric stereo is easily implemented. The stereo computation, after an initial calibration step, is purely local and may be implemented by table lookup, allowing real-time performance. Photometric stereo is a practical scheme for environments, such as industrial inspection, in which the nature and position of the incident illumination can be controlled. [25]

Image Analysis and Scene Analysis

One thing that seems to make computational vision difficult is the fact that many different kinds of knowledge can influence the interpretation of an image. Humans, for example, are good at recognizing the identity of a familiar face in a photograph. The step from image to interpretation seems immediate. Intermediate stages seem beyond introspection.

An important question to ask concerns how much of the interpretation is forced by the image and how much can be attributed to the influence of prior knowledge and expectation. This is a difficult question in human perception because it is impossible to separate one effect from the other. At best, one might characterize human vision as the interaction of diverse, partially complete and highly redundant knowledge sources.

In computational vision, it has been useful to make the distinction between image analysis and scene analysis. The purpose of image analysis is to produce a symbolic description of the "important" intensity changes in an image. (Needless to say, it is difficult to define "important"). The purpose of scene analysis is to produce a symbolic description of scene features according to some externally defined goal.

The exact boundary is often a matter of taste. For some, it is the distinction between bottom-up (data driven) and top-down (model driven) interpretation. For others, it is the distinction between domain independent and domain specific interpretation. It seems, however, that both image analysis and scene analysis require generic knowledge of imaging and the scene domain. This knowledge includes properties of sensors, laws of physical optics and constraints on possible configurations in the world.

While the exact boundary is unnecessary to define, it is nevertheless clear that it has moved. One consequence of early work in the blocksworld was the view that descriptions produced by image analysis were fundamentally impoverished. Emphasis shifted to obtaining additional constraint by the downward flow of knowledge from the scene domain. Current work demonstrates that image analysis is capable of producing rich and full descriptions by the upward flow of constraint at levels analogous to the primal sketch, the raw 2 1/2-D sketch, the full 2 1/2-D sketch and 3-D models. Image irradiance measurements carry a great deal of useful information about the underlying object scene. Image analysis attempts to exploit this fact.

A Case for Image Analysis (The Blocksworld Revisited)

In blocksworld vision, the distinction between image analysis and scene analysis seemed clear. The purpose of image analysis was to produce a line drawing of the underlying scene. That is, image analysis converted the array of image irradiance measurements into a symbolic description equivalent to a list of 4-tuples, where each 4-tuple represented the coordinates of the two end points of a line segment. The purpose of scene analysis was to interpret the line drawing in terms of the underlying objects and relationships between objects in the scene. That is, scene analysis grouped line segments into regions and regions into bodies, determined the identity of the objects and determined relationships between objects and groups of objects.

Most of the published literature in blocksworld vision is in the domain of scene analysis [1-8]. That is, one presumes from the outset to start with a line drawing. (The argument that people do very well at interpreting line drawings is not relevant here. The problem, defined for computational vision, is to go from images to surfaces). It was of minor concern that programs to produce the required line drawing were of modest success at best.

It is instructive to examine why simple line-finding programs fail. First, not all edges in the scene produce intensity changes in the image. Depending on the orientation of the planes which intersect and on the position of the light source, image irradiance may be constant across the edge. Second, edges in the scene produce different kinds of intensity changes in the image. Intensity changes vary in sign, magnitude and spatial extent. Often, intensity measurements across an edge show values outside the range of values determined by the planes which intersect to form the edge. Thus, finding line segments with a simple "edge-detection" operator fails. Instead, most programs embody multiple operators, one optimal for each kind of intensity change that has been catalogued.

In early work, no attempt was made to interpret the different kinds of intensity changes. One was never sure whether the image was responding to some physical characteristic of the scene or merely demonstrating some artifact of the sensor.

In current work, we can do more. First, the quality of image sensors is now high enough that we can be sure that the image is responding to physical characteristics of the scene. Second, using the reflectance map and the observation that surface orientation is rarely discontinuous in the real world (i.e., there is likely to be a slight rounding of the edge where two planes meet) one can interpret intensity profiles across lines in terms of the underlying edges semantics (i.e., convex, concave, obscuring, crack, shadow, etc.). A paper by Horn includes an excellent discussion of edge imperfections [10]. Horn presents his results modestly without drawing conclusions beyond the work actually done. I, however, would like to pose the following as an open question:

> The problem of producing a line drawing from the image of a blocksworld scene is the same as the problem of producing a labelled line drawing (i.e., a line drawing for which the underlying edge semantics - convex, concave, obscuring, crack, shadow - has been identified).

It seems that in order to produce any line drawing it is necessary to have already determined the underlying edge semantics. In retrospect, it seems that the line drawing is an impoverished description. It is ridiculous to describe a variety of intensity changes and then throw this description away in order to begin with an unlabelled line drawing. At the very least, image analysis can produce a partially constrained labelling of the line drawing if it can produce a line drawing at all. The conclusion that emerges is that image analysis ought to produce as rich a description as possible of the intensity changes in an image. Assertions about the presence of intensity changes is crucial. But, so are assertions about the size, magnitude and spatial extent of those changes. The research strategy that emerges is to exploit as much as possible the information contained in image intensities. That is not to say that scene analysis is neither necessary nor important. Rather, it argues that a boundary between image analysis and scene analysis cannot be drawn until one knows exactly what can or cannot be determined directly from image intensities.

Conclusions

Computational vision has developed its own paradigm for research. A better understanding is emerging concerning the representations required at different levels of visual processing and concerning the specific knowledge sources and constraints that contribute to the descriptions produced at each level.

At the same time, computational vision continues to resort to stylized worlds for its development. The image irradiance equation, based on the reflectance map and the gradient space, is a powerful theoretical tool indicative of the growing maturity of the field. Research issues can be identified and studied in isolation, in the right stylized world. Difficulties that arise in stylized worlds will certainly be present in more complicated real-worlds. In practical applications, such as industrial automation or remote sensing, it is important to be able to estimate the nature and magnitude of these difficulties and to suggest configurations in which they are minimized.

Acknowledgement

The author's view of computational vision is rooted in his experience as a graduate student and research scientist at the M.I.T. Artificial Intelligence Laboratory. The author acknowledges the influence of B.K.P. Horn, D. Marr and P.H. Winston. The author would like to thank A.K. Mackworth for demonstrating that excellence in computational vision is not restricted to M.I.T. The author also thanks the faculty, staff and

graduate students of the U.B.C. Laboratory for Computational Vision for helping to refine many of the ideas presented in this paper. N.M. Holm prepared the figures (used in the talk but not reproduced in the paper) and L. Wey typed the manuscript).

The author acknowledges the financial support provided by NSERC (grants A3390 and SMI-51), and the U.B.C. Program of Excellence in Remote Sensing.

References

[1] Roberts, L.G., "Machine Perception of Three-Dimensional Objects", in Optical and Electro-Optical Information Processing, J.T. Tippet et al. (eds.), M.I.T. Press, pp. 159-197, 1965.

[2] Guzman, A., "Decomposition of a Visual Scene into Three-Dimensional Bodies", AFIPS Proc. Fall Joint Computer Conf. (33) 291-304, (1968).

[3] Clowes, M.B., "On Seeing Things", Artificial Intelligence (2) 79-112, (1971).

[4] Huffman, D.A., "Impossible Objects as Nonsense Sentences", in Machine Intelligence 6, B. Meltzer & D. Michie (eds.), Edinburgh University Press, pp. 295-323, (1971).

[5] Falk, G., "Interpretation of Imperfect Line Data as a Three-Dimensional Scene", Artificial Intelligence (3) 101-144, (1972).

[6] Mackworth, A.K., "Interpreting Pictures of Polyhedral Scenes", Artificial Intelligence (4) 121-137, (1973).

[7] Winston, P.H., "The M.I.T. Robot", in Machine Intelligence 7, B. Meltzer & D. Michie (eds.), Edinburgh University Press, pp. 431-463, (1973).

[8] Winston, P.H., The Psychology of Computer Vision, McGraw-Hill, (1975).

[9] Marr, D., VISION: A Computational Investigation in the Human Representation and Processing of Visual Information, W.H. Freeman, (1981).

[10] Horn, B.K.P., "Understanding Image Intensities", Artificial Intelligence (8) 201-231, (1977).

[11] Barrow, H.G. & J.M. Tenenbaum, "Recovering Intrinsic Scene Characteristics from Images", in Computer Vision Systems, A.R. Hanson & E.M. Riseman (eds.), pp. 3-26, Academic Press, 1978.

[12] Horn, B.K.P., "Sequins and Quills: Representations for Surface Topography" in Representation of 3-Dimensional Objects, R. Bajcsy (ed.), Springer, (in press).

[13] Grimson, W.E.L., From Images to Surfaces, M.I.T. Press, 1981.

[14] Baker, H.H. & T.O. Binford, "Depth from Edge and Intensity Based Stereo", Proc. IJCAI-81, pp. 631-636, Vancouver, 1981.

[15] Woodham, R.J., "Analyzing Images of Curved Surfaces", Artificial Intelligence (17) 117-140, (1981).

[16] Woodham, R.J., "Photometric Method for Determining Surface Orientation from Multiple Images", Optical Engineering (19), 139-144, (1980).

[17] Ullman, S., The Interpretation of Visual Motion, M.I.T. Press, 1979.

[18] Witkin, A.P., "Recovering Surface Shape and Orientation from Texture", Artificial Intelligence (17) 17-45, (1981)

[19] Marr, D., "Analysis of Occluding Contour", Proc. R. Soc. Lond. B (197) 441-475, (1977).

[20] Stevens, K.A., "The Visual Interpretation of Surface Contours", Artificial Intelligence (17) 47-73, (1981).

[21] Barrow, H.G. and J.M. Tenenbaum, "Interpreting Line Drawings as Three-Dimensional Surfaces", Artificial Intelligence (17) 75-116, (1981).

[22] Horn, B.K.P. and B.G. Schunck, "Determining Optical Flow", Artificial Intelligence (17) 185-203, (1981).

[23] Woodham, R.J., "Using Digital Terrain Data to Model Image Formation in Remote Sensing", Proc. Soc. Photo-Optical Instrumentation Engineers, (238) 361-369, (1980).

[24] Horn, B.K.P. and B.L. Bachman, "Using Synthetic Images to Register Real Images with Surface Models", CACM (21) 914-924, (1978).

[25] Ikeuchi, K., "Determining Surface Orientations of Specular Surfaces by Using the Photometric Stereo Method", IEEE-PAMI (3) 661-669 (1981).

AUTHOR INDEX

Brachman, Ronald J. 212
Bradshaw, Gary 137
Bresina, J.L. 12
Brouse, Roger A. 27
Bryant, Edwin 48
Cercone, Nick 172
Chandrasekaran, B. 222
Covvey, H. Dominic 71
Davidson, Jim 204
Davies, D. Julian M. 158
Douglass, Robert J. 92
Elcock, E.W. 121
Fahlman, Scott E. 230
Furley, Arthur M. 8
Funt, Brian 48
Gilmore, John F. 60
Glickman, Jay 33
Goebel, Randy 172
Gordon, Richard 41
Gullen, Andrew 108
Hegner, Stephen J. 92
Kautz, Henry A. 19
Korein, James U. 79
Kountanis, Dionysios 132
Kramer, Bryan M. 108
Langley, Pat 137
Lesperance, Yves 106
Levine, Martin D. 52
Maier, David 196
McCalla, Gordon 65
Mercer, R.E. 103
Meyers, Hugh 108
Mittal, Sanjay 164
Mylopoulos, John 71, 108
Nazif, Ahmed M. 52
Papalaskaris, Mary Angela .. 97
Patel-Schneider, Peter F. . 108
Peachey, Darwyn 85
Rangaraj, M.R. 41
Reiter, R. 103
Rendell, Larry A. 150
Robertson, Paul 144
Salveter, Sharon C. 196
Schank, Roger C. 236
Schubert, Lenhart K. .. 97, 189
Shibhara, Tetsutaro 71
Shinghal, Rajjan 66
Simon, Herbert A. 137
Sridharan, N.S. 12
Tsotsos, John K. 71
Ward, Blake 85
Wilkins, David E. 1
Woodham, Robert J. 237
Zadeh, Lotfi A. 116
Zarri, Gian Piero 183