AN OVERVIEW OF KNOWLEDGE REPRESENTATION[†]

John Mylopoulos
Department of Computer Science
University of Toronto

1. INTRODUCTION

This is a brief overview of terminology and issues related to Knowledge Representation (hereafter KR) research, intended primarily for researchers working on Semantic Data Models within Database Management and Program Specifications within Programming Languages/Software Engineering.

Knowledge Representation is a central problem in Artificial Intelligence (AI) today. Its importance stems from the fact that the current design paradigm for "intelligent" systems stresses the need for expert knowledge in the system along with associated knowledge-handling facilities. This paradigm is in sharp contrast to earlier ones which might be termed "power-oriented" [Goldstein and Papert 77] in that they placed an emphasis on general purpose heuristic search techniques [Nilsson 71].

The basic problem of KR is the development of a sufficiently precise notation for representing knowledge. Following [Hayes 74] we shall refer to any such notation as a (knowledge) *representation scheme*. Using such a scheme one can specify a *knowledge base* consisting of *facts*. For the purposes of this paper, a knowledge base will be treated as a model of a world/enterprise/slice of reality.[††]

There already exist a number of important papers on the subject. [Hayes 74] deals with central issues of KR theory, while [Bobrow and Collins 75] includes a fine collection of papers on KR theory and practice. More recently, [Findler 79], [Waterman & Hayes-Roth 79], and [Gallaire & Minker 78] present collections of papers on semantic networks, production systems and logical representation schemes respectively. [Bobrow 77] contains an interesting collection of short presentations on a number of state-of-the-art schemes while [Goldstein & Papert 77] relates KR to other important problems in AI. A recently published SIGART Newsletter issue [Brachman & Smith 80] contains questionnaire results from more than 80 research groups working on or using a representation scheme. An extended survey of KR research is available in [Barr & Davidson 80]. Finally, [Wong and Mylopoulos 77] examines KR issues and searches for counterparts in Database Management.

2. A TAXONOMY OF REPRESENTATION SCHEMES

Representation schemes have been classified into *declarative* and *procedural* ones [Winograd 75]. For the purposes of the discussion that follows, we further subdivide declarative schemes into *logical* and (semantic) *network* ones. While reading the rest of the section, the reader should keep in mind that very few representation schemes fit cleanly into one of the three categories as most of them combine features from more than one category. A particular class of such schemes based on Minsky's *frame proposal* [Minsky 75] is discussed separately.

2.1 *Logical Representation Schemes*

Such schemes employ the notions of constant, variable, function, predicate, logical connective and quantifier to represent facts as logical formulas in some logic (First or Higher Order/Multi-valued/Modal/Fuzzy etc.). A knowledge base, according to this view, is a collection of logical formulas which provides a partial description of a world. Modifications to the knowledge base occur with the introduction/deletion of logical formulas so logical formulas are the atomic units for knowledge base manipulation in such schemes. The use of Logic as a representation scheme can be traced at least as far back as McCarthy's "Advice Taker" [McCarthy 68].

An important advantage of logical representation schemes is the availability of inference rules in terms of which one can define proof procedures. Such procedures can be used for information retrieved [Reiter 78a], semantic constraint checking [Nicolas and Yazdanian 78] and problem solving [Green 69]. [Nilsson 71] presents a review of

† Prepared for the Workshop on Data Abstraction, Databases and Conceptual Modelling, Pingree Park, Colorado, June 23-26, 1980.
†† For other ways of viewing a knowledge base see [Brachman & Smith 80, p.68].

early results, applications and promises of theorem-proving research while [Gallaire and Minker 78] present more recent work on logical schemes, theorem-proving and their applications to Database Management.

Another strength of logical schemes is the availability of a clean, well-understood and well-accepted formal semantics [Mendelson 64], at least for "pure" logical schemes that are quite close to First Order Logic. As one moves to representation schemes that try to deal with knowledge acquisition [McDermott and Doyle 78], beliefs [Moore 77] and defaults [Reiter 78b] the availability of a clean formal semantics becomes more problematic and is an area of active research.

A third strength of logical schemes is the simplicity of the notation employed which leads to knowledge base descriptions that are understandable. Another advantage is the conceptual economy encouraged by logical representation schemes which allow each fact to be represented once, independently of its different uses during the course of its presence in the knowledge base.

A major drawback of logical schemes is the lack of organizational principles for the facts constituting a knowledge base. A large knowledge base, like a large program, needs organizational principles to be understandable as a unit. Without them, a knowledge base can be as unmanageable as a program written in a programming language which does not support abstraction facilities.

A second drawback is the difficulty in representing procedural and heuristic knowledge† such as

"If you are trying to do A while condition B holds, try strategies $C_1, C_2, \ldots, C_n$".

An interesting departure from logical representation schemes has been proposed by Kowalski [Kowalski 74] who argues in favour of a dual semantics for logical formulas of the form

$$B_1 \wedge B_2 \wedge \ldots \wedge B_n \supset A$$

The first is the traditional Tarskian semantics. The second is a procedural semantics which interprets the formula as

"If you want to establish A, try to establish B_1 and B_2 and ... and B_n".

The language PROLOG [Kowalski 74] realizes this idea and has gained many supporters as it combines advantages from logical and procedural representation schemes. FOL [Weyhrauch 78] also attempts to deal with control issues within a logical framework.

Logical schemes are strongly related to Codd's Relational Model [Codd 70] so it is fair to argue that such schemes have their counterparts in Database Management.

† [Hayes 77] argues against this point.

2.2 *Network Representation Schemes*

Such schemes, often called *semantic networks*, attempt to describe a world in terms of objects (nodes) and binary associations (*labelled edges*), the former denoting individuals and the latter binary relationships in the world being modelled. According to a network representational view, a knowledge base is a collection of objects and associations, or a directed labelled graph, and modifications to the knowledge base occur through the insertion/deletion of objects and the manipulation of associations. Semantic networks were first used by [Quillian 68] and [Raphael 68] and have gained wide acceptance as means of modelling human memory (e.g. [Norman and Rumelhart 75]) and as useful representations for building "intelligent" systems (e.g. [Walker 76]).

Early versions of network schemes tended to encourage a proliferation of association types (edge labels) as new kinds of knowledge were represented. This practice and other deficiencies of earlier network schemes have been criticized in [Woods 75] and [Schubert 76]. Their criticisms have triggered a trend towards network schemes with a fixed number of primitive association types which have well-defined semantics and are descriptively adequate in that they can be used to represent any fact expressible in a logical scheme. Some of these schemes simply view network knowledge bases as convenient implementations of logical ones ([Shapiro 79],[Schubert 76], etc.). Others, notably KLONE [Brachman 79], view network schemes as tackling a different set of representational issues and propose a set of primitive association types accordingly.

Some of the primitive association types that have been used are discussed briefly below:

Classification (MEMBER-OF/INSTANCE-OF)

This association type relates an object (e.g. john-smith) to its generic type(s) (e.g. PERSON, MALE, STUDENT). Inclusion of this association type in a network scheme forces a distinction between objects (*tokens*) and object types. Some network schemes use classification recursively to define object (meta) types with instances other object types (e.g. [Levesque and Mylopoulos 79]).

Aggregation (PART-OF)

This principle relates an object (e.g. john-smith) to its components. For example, the parts of John Smith, viewed as a physical object, are his head, arms, etc.; when viewed as a social object, they are his address, social insurance number etc. As with classification, aggregation can be applied recursively so that one can represent the components of the components of an object etc.

Generalization (IS-A/SUBSET-OF)

It relates an object type (e.g. STUDENT) to a more generic one (e.g. PERSON). It is particularly useful in situations where there is a proliferation of object types as it organizes them into a hierarchy according to their generality/specificity.

Contexts (Partitions)

Another method of organizing network knowledge bases is proposed in [Hendrix 75] and involves grouping objects and associations into *partitions* which are organized hierarchically so that if partition A is below partition B, everything visible or present in B is also visible in A unless otherwise specified. Partitions have been found useful in representing hypothetical worlds and belief spaces (e.g. [Cohen 78]).

Not all network schemes offer the association types mentioned above. For example, [Fahlman 79] uses one association type to represent both classification and generalization.

Due to their nature, network schemes address directly issues of information retrieval since associations can be used to define access paths for traversing a network knowledge base. Another important feature of network schemes is the potential use of primitive association types such as those mentioned above for the organization of a knowledge base. A third advantage is the obvious graphical representation of network knowledge bases which enhances their understandability.

A major drawback of network schemes has been the lack of a formal semantics and a standard terminology. This is at least partly due to the fact that semantic networks have been used as representational tools in very different ways. For example, [Schubert 76] uses them to represent logical formulas, [Brachman 79] to organize knowledge, [Schank 75] to represent the "meaning" of natural language sentences and [Szolovits et al. 77] to represent linguistic expressions.

Semantic networks share some common characteristics with the DBTG data model [CODASYL DBTG 1971] and are even more closely related to various object-oriented data models proposed more recently (e.g. [Hall et al. 76]).

2.3 *Procedural Representation Schemes*

Such schemes view a knowledge base as a collection of procedures expressed in some language. Most procedural schemes have been influenced quite heavily by LISP which has been used almost exclusively as the implementation language for "intelligent" systems. Indeed, in the past LISP itself was a favorite representation scheme due to, among other things, its purely symbolic nature and the dynamic run-time environment it offers its users.

Procedural schemes beyond LISP can be classified on the basis of the stand they take with respect to two issues. The first is concerned with the activation mechanism offered for procedures, while the second involves the control structures offered by any one scheme.

On the first issue, PLANNER [Hewitt 71,72] introduced the notion of *pattern directed procedure invocation*. A knowledge base is viewed in PLANNER as a global database of assertions and a collection of *theorems* (or *demons*) which watch over it and are activated whenever the database is modified or searched. Each theorem has an associated *pattern* which, upon the theorem's activation, is matched against the data about to be inserted/removed or retrieved from the database. If the match succeeds, the theorem is executed. Thus with theorems the usual procedure calling mechanism is replaced with one where procedures are called whenever a condition is satisfied.

Production systems [Waterman and Hayes-Roth 79] offer a procedural scheme that is in many ways similar to PLANNER. A knowledge base is a collection of *production rules* and a global database. Production rules, like theorems, consist of a pattern and a body involving one or more *actions*. The database begins in some initial state and rules are tried out in some prespecified order until one is found whose pattern matches the database. The body of that rule is then executed and matching of other rules continues.†

There are major differences between the activation mechanism of a PLANNER theorem and a production system rule as well. The order in which theorem patterns are matched is undetermined in PLANNER (although the user can define one for any particular situation where he tries to tamper with the database). "Standard" production systems, like Markov algorithms, have a fixed ordering of rules which determines when will each rule be matched against the database. Another important difference is that theorems can call directly other theorems while productions can only do so indirectly by placing appropriate information on the database. Thus, a production system database can be viewed as a workspace or a bulletin board which provides the only means of communication between rules.

Turning to control structures, there exist several proposals which extend or otherwise modify the usual hierarchical control structure of LISP or ALGOL. As indicated in the previous paragraph, production systems offer one where there is no direct communication or control between rules. Thus a production system knowledge base consists of a collection of *loosely coupled* rules and this feature renders such knowledge bases fairly easy to understand and modify.

PLANNER's control structure for theorems uses *backtracking* in that when a theorem's body is executed and fails to achieve a predetermined goal, the side-effects of the unsuccessful theorem are erased and other theorems are tried until one is found that succeeds. It has been argued quite convincingly that backtracking is an unwieldy control structure [Sussman and McDermott 72].

An extreme proposal as far as control structures are concerned is Hewitt's ACTOR formalism [Hewitt et al. 73,74] which views *all* objects that are part of a knowledge base as *actors*, i.e. active agents which play a role on cue according to a script. Actors are capable of sending an receiving *messages* which, naturally, are also actors. Thus writing a program in the ACTOR formalism involves deciding on the objects in the domain, the messages each object should receive and what each object

† The account presented here is an idealization of production systems as most of them vary on the form of rules and the order in which they are tried [Davis and King 75].

should do when it receives each kind of message. The ACTOR formalism basically does not impose a preconceived control structure on its user. Instead, it provides him with control primitives so that he can define his own.

Procedural schemes have in principle one major advantage and one major disadvantage compared to declarative ones. They allow the specification of direct interactions between facts thus eliminating the need for wasteful searching [Winograd 75]. On the other hand, a procedural knowledge base, like a program, is difficult to understanding and modify. Each of the proposed schemes discussed in the previous paragraphs goes some distance towards eliminating the disadvantages of pure procedural schemes while maintaining their advantages.

2.4 *Frame-based Representation Schemes*

Since 1975, when Minsky originally proposed it ([Minsky 75]), the notion of *frame* has played a key role in KR research. A frame is a complex data structure for representing a stereotypical situation such as being in a certain kind of living room or going to a child's birthday party. The frame has slots for the objects that play a role in the stereotypical situation as well as relations between these objects. Attached to each frame are different kinds of information such as how to use the frame, what to do if something unexpected happens, default values for its slots etc. A knowledge base is now a collection of frames organized in terms of some of the organizational principles discussed earlier but also other "looser" principles such as the notion of *similarity* between two frames.

The original frame proposal was nothing but a framework for developing representation schemes which combined ideas from semantic networks, procedural schemes, linguistics etc. Several representation schemes proposed since then have adapted the frame proposal. Below we present brief descriptions for four of them.

FRL [Goldstein and Roberts 77]

An FRL knowledge base consists of frames whose slots carry information such as comments on the source of a value bound to the slot, a default value, constraints, and procedures that are activated when a value is bound, unbound or needed for a slot. All frames are organized into a hierarchy which appears to be a combination of classification and generalization. The procedures attached to a slot are expressed in LISP.

KRL [Bobrow and Winograd 77]

This is a more ambitious project than FRL. Like FRL, the basic units of a KRL knowledge base are frames with slots and several kinds of information attached to each slot. Unlike FRL where this information provides details about how to instantiate a frame, KRL is much more concerned with a matching operation for frames. All on-going processes at any one time are controlled through a multiprocessor agenda which can be scheduled by the designer of the knowledge base. KRL also supports belief contexts which can serve to define an attention focusing mechanism. "Self knowledge" can be included in a knowledge base by providing description about other descriptions.

OWL [Szolovits et al. 77]

Unlike other frame-oriented schemes, OWL bases its features on the syntactic and semantic structure of English, taking as founding principle the Whorfian Hypothesis that a person's language plays a key role in determining his model of the world and thus in structuring his thought. An OWL knowledge base can be viewed as a semantic network whose nodes are expressions representing the meaning of natural language sentences. Each node, called a concept, is defined by a pair (genus, specializer) where "genus" specifies the type or superconcept while "specializer" serves to distinguish this concept from all other concepts with the same genus.

KLONE [Brachman 79]

A KLONE knowledge base is a collection of concepts where each concept is a highly structured object, having slots to which one can attach a variety of information (defaults, modalities etc.). To a concept one can also attach structural descriptions which express constraints on the values that can be bound to the different slots of the concept. Concepts provide purely descriptional structure and make no assertions about existence of a referent or coreference of descriptions. A separate construct called a *nexus* is used to make assertions about the world being modelled. Also, KLONE offers procedural attachment as a means of associating procedural information, expressed at this time in LISP, with a concept. Another important feature of KLONE is the strong organization of concepts it encourages through a version of the IS-A association type discussed in section 2.2.

3. DISTINGUISHING FEATURES OF REPRESENTATION SCHEMES

The reader with a background in Database Management and/or Programming Languages must have already noticed the similarity in basic goals between KR research as we have described in this paper and research on Semantic Data Models or Program Specifications. In all three cases the aim is to provide tools for the development of formal descriptions of a world/enterprise/slice of reality. The tools under consideration involve a representation scheme/semantic data model/specification language which serves as the linguistic vehicle for such descriptions. A common demand on representation schemes developed in all three areas is that descriptions expressed in terms of the scheme be "natural". Below we list some of the more technical (and less vague) characteristics of representation schemes which appear to distinguish them from their semantic data model/program specification language cousins.

3.1 *Multiple Uses of Facts*

Unlike a database, whose facts are used almost exclusively for retrieval purposes or a program whose facts are used in the execution of some procedure, a knowledge base contains facts which may have multiple uses. A representation scheme must take this into account in terms of the tools it offers. Below we list some possible uses ([Bobrow 75]).

Inference

Given a collection of facts, new facts may be deduced from them according to some fixed rules of inference without interaction with the outside world. Some inferences have the flavour of inference techniques in formal logic. For knowledge bases, however, it is also useful sometimes to derive facts through specialized procedures that use other known facts only in fixed ways. For example, a procedure that determines whether a pair is in the transitive closure of some binary relation can perform inferences of a very specialized nature and is only applicable to facts associated with a transitive relation. Also, a knowledge base may be represented in such a way that there are "preferred inferences". The use of defaults is a good example of such a mechanism.

Deduction, with a formal, special purpose or heuristic flavour, is not the only kind of inference. There can also be inductive inferences [Brown 73] and abductive ones [Pople 73] which have played a role in some knowledge bases.

Given all this variety for inference mechanisms, the question for the designer of a representation scheme is not how he can include all of them in his scheme, but which ones, if any, he is going to include. Logical schemes clearly have an advantage over other types of schemes when considered from the point of view of (general purpose) inference facilities.

Access

Access (and storage) of information in a knowledge base for question-answering purposes constitutes an all-important use of the knowledge base. The associationist viewpoint of network schemes, particularly their organizational principles, make them strong candidates for access-related uses.

Matching

Matching as a knowledge base operation can be used for a variety of purposes, including (i) *classification*, i.e. determining the type of an unknown input, (ii) *confirmation* where a possible candidate to fit a description is matched against it for confirmation purposes, (iii) *decomposition* where a pattern with a substructure is matched against a structured unknown and the unknown is decomposed into subparts corresponding to those of the pattern, (iv) *correction* where the nature of a pattern match failure leads to error correction of the unknown input.

The matching operation itself can be (i) *syntactic* where the form of the unknown input is matched against another form, (ii) *parametric* in the tradition of Pattern Recognition research [Duda and Hart 73], (iii) *semantic* where the function of the components of the pattern is specified and the matcher attempts to find elements of the input to serve this function, (iv) *forced* as in MERLIN [Moore and Newell 74] where a structure is viewed as though it were another and matches of corresponding items may be forced.

KRL has paid special attention to matching as a knowledge base operation.

3.2 *Incompleteness*

Except for situations where a knowledge base models artificial "microworlds" (e.g. [Winograd 72]), it cannot be assumed that the knowledge base is a complete description of the world it is intended to model. This observation has important consequences for the operations defined over a knowledge base (inference, access, matching) as well as the design methodologies for knowledge bases.

Consider first operations on a knowledge base. Incompleteness of the knowledge base can lead to very different answers to questions such as

"Is there a person who lives in Toronto?",

depending on whether it is assumed that the persons currently represented in the knowledge base are the only persons in the world being modelled. Similarly, from the facts

"Someone is married to Mary"
"John is not married to Mary"

one can draw different conclusions if George is the only other person represented in the knowledge base. Similar remarks apply for matching.

Until recently much of the work on KR ignored the problem of incompleteness or dealt with it in an ad hoc way. Recent work by [Reiter 78b] and [Levesque 79], among others, attempts to correct this situation.

Viewing a knowledge base as an incomplete and approximate model of a world which can always be improved but can never be quite complete, leads to design methodologies for knowledge bases which are drastically different from ones for programs. Thus in Programming Language the leading design methodologies stress "once and for all" designs where the designer sits down with a clear idea of the algorithm he wants to realize and by the time he stands up the design is complete [Wirth 71]. In AI, a knowledge base is developed over a period of time that can be as long as its lifetime through different *knowledge acquisition* processes that can range from interactive sessions with an expert (e.g. [Davis 77]) to the automatic generation of new facts based on the system's "experiences" (e.g. [Lenat 77]). Organizational principles underlying the structure of a knowledge base can play a crucial role in determining the direction of knowledge acquisition, i.e. which facts should be acquired first and which ones later.

3.3 *Self Knowledge*

There are many kinds of self knowledge. Facts which describe the form or allowable configurations of other facts (e.g. type definitions) are an important kind of self knowledge. Making such facts available for question answering and inference by representing them the same way as other facts is an important capability of declarative schemes which is generally not shared by procedural ones. A good example of use of such self knowledge for knowledge acquisition is provided in TEIRESIAS [Davis 77].

A second kind of self knowledge involves the ability of a system to answer elementary questions about its actions as in SHRDLU [Winograd 72], or about the strategies it uses to perform some task as in HACKER [Sussman 75].

4. CONCLUSIONS

There are signs today that KR is maturing at least to the point where there is some agreement on issues and open questions.† One can find several knowledge-based systems which perform at an expert or near expert level ([Feigenbaum 77]). There is even some discussion on issues related to *Knowledge Engineering* ([Feigenbaum 77]) which appears to suggest that design methodologies for knowledge bases are following a similar path as design methodologies for large programs, perhaps with a 8-10 year lag (see Software Engineering developments since 1968).

ACKNOWLEDGEMENTS

Alex Borgida and Hector Levesque read an earlier draft of this paper and offered many thoughtful suggestions for improvements.

† The extent of this agreement is outlined quite well in the responses to a questionnaire on KR research ([Brachman and Smith 80]).

References

[1] [Barr and Davidson 80]
Barr, A. and Davidson, J., "Representation of Knowledge" in *Handbook of Artificial Intelligence*, Barr, A. and Feigenbaum, E. (eds.) Computer Science Dept. Report No. STAN-CS-80-793, Stanford University, 1980.

[2] [Bobrow 75]
Bobrow, D., "Dimensions of Representation" in [Bobrow and Collins 752.

[3] [Bobrow and Collins 75]
Bobrow, D. and Collins, A. (eds.), *Representation and Understanding*, Academic Press, 1975.

[4] [Bobrow 77]
Bobrow, D., "A Panel on Knowledge Representation", Proceedings IJCAI-77, Cambridge, Mass., August 1977.

[5] [Bobrow and Winograd 77]
Bobrow, D. and Winograd, T., "An Overview of KRL, A knowledge Representation Language", Cognitive Science, vol.1, no.1, January 1977.

[6] [Brachman 79]
Brachman, R., "On the Epistemological Status of Semantic Networks" in [Findler 79].

[7] [Brachman and Smith 80]
Brachman, R. and Smith, B., "Special Issue on Knowledge Representation", SIGART, no.50, Feb. 1980.

[8] [Brown 73]
Brown, J.S., "Steps Towards Automatic Theory Formation", Proceedings IJCAI-73, Palo Alto, CA, August 1973.

[9] [CODASYL DBTG 71]
CODASYL DBTG Report, Conf. Data Sys. Languages, ACM, New York, 1971.

[10] [Codd 77]
Codd, E.F., "A Relational Model of Data for Large Shared Data Banks", Comm. ACM, vol.13, 377-387.

[11] [Cohen 78]
Cohen, P.R., *On Knowing What to Say: Planning Speech Acts*, Ph.D. thesis, Dept. of Computer Science, Univ. of Toronto, 1978; also TR-118.

[12] [Davis 77]
Davis, R., "Interactive Transfer of Expertise: Acquisition of New Inference Rules", Proceedings IJCAI-77, Cambridge, Mass., August 1977.

[13] [Davis and King 75]
Davis, R. and King, J., "An Overview of Production Systems", Memo AIM-271, Stanford Artificial Intelligence Laboratory, 1975.

[14] [Duda and Hart 73]
Duda, R.O. and Hart, P.E., *Pattern Classification and Scene Analysis*, Wiley-Interscience, 1973.

[15] [Fahlman 79]
Fahlman, S.E., *NETL: A System for Representing and Using Real-World Knowledge*, MIT Press, 1979.

[16] [Feigenbaum 77]
Feigenbaum, E.A., "The Art of Artificial Intelligence: Themes and Case Studies of Knowledge Engineering", Proceedings IJCAI-77, Cambridge, Mass., August 1977.

[17] [Findler 79]
Findler, N.V., *Associative Networks: Representation and Use of Knowledge by Computer*, Academic Press, 1979.

[18] [Gallaire and Minker 78]
Gallaire, H. and Minker, J., *Logic and Databases*, Plenum, 1978.

[19] [Goldstein and Papert 77]
Goldstein, I. and Papert, S., "Artificial Intelligence, Language and the Study of Knowledge", Cognitive Science, vol.1, no.1, 1977.

[20] [Goldstein and Roberts 77]
Goldstein, I. and Roberts, R.B., "NUDGE: A Knowledge-Based Scheduling Program", Proceedings IJCAI-77, Cambridge, Mass., August 1977.

[21] [Green 69]
Green, C., *The Application of Theorem Proving to Question-Answering Systems*, Ph.D. thesis, Dept. of Electrical Engineering, Stanford University, 1969.

[22] [Hall et al. 76]

Hall, P., Owlett, J., Todd, S.,"Relations and Entities" in *Modelling in Database Management Systems*, Nijssen, G. (ed.), North-Holland Pub. Co., 1976.

[23] [Hayes 74]
Hayes, P.J., "Some Problems and Non-Problems in Representation Theory", Proceedings AISB Summer Conference, Essex University, 1974.

[24] [Hayes 77]
Hayes, P.J., "In Defence of Logic", Proceedings IJCAI-5, Cambridge, Mass., 559-565, 1977.

[25] [Hendrix 75]
Hendrix, G., "Expanding the Utility of Semantic Networks Through Partitioning", Proceedings IJAI-75, Tbilisi USSR, Sept. 1975.

[26] [Hewitt 71]
Hewitt, C., "PLANNER: A Language for Proving Theorems in Robots", Proceedings IJCAI-71, London, England, August 1971.

[27] [Hewitt 72]
Hewitt, C., *Description and Theoretical Analysis (Using Schemata) of PLANNER: A Language for Proving Theorems and Manipulating Models in a Robot*, Ph.D. thesis, Dept. of Mathematics, MIT, 1972.

[28] [Hewitt et al. 73]
Hewitt, C., Bishop, P., Steiger, R., "A Universal Modular Actor Formalism for Artificial Intelligence", Proceedings IJCAI-73, Palo Alto, CA, August 1973.

[29] [Hewitt et al. 74]
Hewitt, C., Greif, I., "Actor Semantics of PLANNER-73", Working Paper No.81, MIT AI Laboratory, 1974.

[30] [Kowalski 74]
Kowalski, R., "Predicate Logic as a Programming Language", Proceedings IFIP Congress, 1974, 569-574.

[31] [Lenat 77]
Lenat, D.B., "The Ubiquity of Discovery", Proceedings IJCAI-77, Cambridge, Mass., August 1977.

[32] [Levesque 79]
Levesque, H., "The Representation of Incomplete Knowledge as Applied to a Logic of Sentences", AI Memo 79-1, Dept. of Computer Science, Univ. of Toronto, 1979.

[33] [Levesque and Mylopoulos 79]
Levesque, H. and Mylopoulos, J., "A Procedural Semantics for Semantic Networks" in [Findler 79].

[34] [McCarthy 68]
McCarthy, J., "Programs with Common Sense" in *Semantic Information Processing*, Minsky, M. (ed.), MIT Press, 1968.

[35] [McDermott and Doyle 78]
McDermott, D., Doyle, J., "Non-monotonic Logic I", AI Memo 486, MIT AI Laboratory, 1978.

[36] [Mendelson 64]
Mendelson, E., *Introduction to Mathematical Logic*, Van Nostrand, 1964.

[37] [Minsky 75]
Minsky, M., "A Framework for Representing Knowledge", in P. Winston (ed.) *The Psychology of Computer Vision*, McGraw-Hill, 1975.

[38] [Moore 77]
Moore, R., "Reasoning About Knowledge and Action", Proceedings IJCAI-77, Cambridge, Mass., August 1977.

[39] [Moore and Newell 74]
Moore, J. and Newell, A., "How can MERLIN Understand?" in L. Gregg (ed.) *Knowledge and Cognition*, Lawrence Erlbaum Assoc., 1974.

[40] [Nicolas and Yazdanian 78]
Nicolas, J.M. and Yazdanian, K., "Integrity Checking in Deductive Databases", in [Gallaire and Minker 78].

[41] [Nilsson 71]
Nilsson, N., *Problem Solving Methods in Artificial Intelligence*, McGraw-Hill, 1971.

[42] [Norman and Rumelhart 75]
Norman, D., Rumelhart, D., and the LNR Research Group *Explorations in Cognition*, Freeman, 1975.

[43] [Pople 73]
Pople, H.E., "On the Mechanization of Abductive Logic", Proceedings IJCAI-73, Palo Alto, CA, August 1973.

[44] [Quillian 68]
Quillian, R., "Semantic Memory" in M. Minsky (ed.) *Semantic Information Processing*, MIT Press, 1968.

[45] [Raphael 68]
Raphael, B., "A Computer Program for Semantic Information Retrieval" in *Semantic Information Processing*, Minsky, M. (ed.), MIT Press, 1968.

[46] [Reiter 78a]
Reiter, R., "Deductive Question-Answering on Relational Databases" in [Gallaire and Minker 78].

[47] [Reiter 78b]
Reiter, R., "On Reasoning by Default", Proceedings TINLAP-2, Univ. of Illinois, Urbana, July 1978.

[48] [Schank 75]
Schank, R., *Conceptual Information Processing*, North Holland, 1975.

[49] [Schubert 76]
Schubert, L., "Extending the Expressive Power of Semantic Networks", Artificial Intelligence, vol.7, no.2, 1976.

[50] [Shapiro 79]
Shapiro, S., "The SNePs Semantic Network Processing System", in Findler 79].

[51] [Sussman 75]

Sussman, G.J., *A Computer Model of Skill Acquisition*, MIT Press, 1975.

[52] [Sussman and McDermott 72]
Sussman, G.J. and McDermott, D., "Why Conniving is Better Than Planning", MIT AI Memo 255A, MIT AI Laboratory, 1972.

[53] [Szolovits et al. 77]
Szolovits, P., Hawkinson, L. and Martin W.A., "An Overview of OWL, A Language for Knowledge Representation", MIT/LCS/TM-86, MIT Laboratory for Computer Science, 1977.

[54] [Walker 76]
Walker, D.E. (ed.) *Speech Understanding Research*, North Holland, 1976.

[55] [Waterman and Hayes-Roth 79]
Waterman, D. and Hayes-Roth, F. (eds) *Pattern-Directed Inference Systems*, Academic Press, 1979.

[56] [Weyhrauch 78]
Weyhrauch, R.W., "Prolegomena to a Theory of Mechanized Formal Reasoning", Stanford AI Lab., Memo. 315, 1978.

[57] [Winograd 72]
Winograd, T., *Understanding Natural Language*, Academic Press, 1972.

[58] [Winograd 75]
Winograd, T., "Frame Representation and the Declarative-Procedural Controversy", in [Bobrow and Collins 75].

[59] [Wirth 71]
Wirth, N., "Program Development by Stepwise Refinement", Comm. ACM, vol.14, no.4, 1971.

[60] [Wong and Mylopoulos 77]
Wong, H.K.T. and Mylopoulos, J., "Two Views of Data Semantics: Data Models in Artificial Intelligence and Database Management", INFOR, vol.15, no.3, 1977.

[61] [Woods 75]
Woods, W.A., "What's in a Link: Foundations for Semantic Networks", in [Bobrow and Collins 75].