Int. J. Man-Machine Studies (1979) **11**, 481–500

Logical foundations for database systems†

BRIAN R. GAINES
Centre for Man–Computer Studies, Colchester, U.K.

Database systems originated as mechanisms for storing and retrieving information. Codd's relational formalism generalized and made far more flexible the forms of data structure and retrieval specification allowed. However, available implementations of relational databases are in terms of hard, static, deterministic relations; whereas in real-world applications data is often imprecise, inherently dynamic and non-deterministic. In recent years there has been a range of developments concerned with representing and using data that can only be represented in these "softer" terms. Some of the work has been explicitly concerned with database systems, but much of it, whilst highly relevant, has been in other application areas.

This paper classifies and surveys work on a variety of logical systems in the context of its relevance to database systems. The objective is to show through illustrations what can be incorporated into relational database systems to allow for a wider range of real-world requirements and closer man–computer interation. The current state-of-the-art in natural language interaction with databases is illustrated and discussed. The possibility of paradoxes leading to oscillations in database states is demonstrated. The roles of modal, multi-valued and fuzzy logics in databases are described as discussed.

1. Introduction

The emergence of database technologies (Nolan, 1973*a*; Sibley, 1976) as distinct applied sciences of computing during the past decade is one of the most significant features of the transition in computer applications: from "data processing" to "information processing" and perhaps eventually to "knowledge processing". This transition is of key importance to those of us who see the possibility of far closer interaction between people and computers than is being achieved at present—some form of man-machine symbiosis (Licklider, 1968) in which the strengths of each partner mutually compensate for the weaknesses and complement the strengths of the other. The original function of the computer, as a "number cruncher", certainly enables it to compensate for man's weakness in arithmetic and logical power, but it is not a sufficiently strong link to the mainstream activities of the mind to lead to meaningful symbiosis. For that we need an interface into man's world of knowledge.

In discussing how man can cope with the increasing complexity of society, De Bono noted recently:

> "The computer will be to the organisation revolution what steam power was to the industrial revolution. The computer can extend our organizing power in the same way as steam extended our muscle power" (De Bono, 1979, p. 18).

This analogy can be an extremely useful one when we come to consider what it is that we really require of computers in order that they may enter into a close working relationship with man—one that actually gives the "amplification of intelligence" so often

† This paper is based on one presented at EUROCOMP 78.

0020-7373/79/040481+20$02.00/0 © 1979 Academic Press Inc. (London) Limited

hoped for but not yet achieved. Throughout his extensive studies of the way in which the acquisition of new knowledge can take place Karl Popper (1976) has emphasized the distinct ontological status of the "knowledge" itself as it becomes separate from man and embodies in written records, libraries, etc. I have suggested that the role of the computer in man's world of knowledge, Popper's "World Three", is best seen as analogous to that of engines in the physical world, Popper's "World One":

> "It brings world 3 into the demense of man just as did the steam, internal combustion, and jet engines, world 1. That is, we can move about in, conquer, control and fabricate to our needs the lands and materials of world 3 in a way that makes our previous efforts, all but a few, look feeble" (Gaines, 1978).

Thus, we may think of database technology as the beginning of a process whereby the computer, as a powerful information-processing "engine", begins to be given access to the actual information structures that are of significance to us. Since the cost of computers has, until the recent advent of "personal computers", made it essential to justify their development in terms of commercial applications it has been these that have dominated database developments so far. In common with all such computer-related "technologies", there is no single advance in hardware, software or fundamental concepts, that led to the inception of databases. Indeed, one can see the strongest drive behind them as a purely normative commercial requirement for information storage and retrieval systems suited to the needs of large, complex organisations. The high-level technology of databases is making demands on the lower-level technologies of processors, discs, terminals, operating systems, languages and communications, that are now major forces in determining the directions of commercial development.

Nolan, whose "stage" hypotheses (Nolan, 1973*b*) have given a highly dynamic and realistic model of the assimilation of data-processing technologies by organizations, has presented a set of possible hypotheses as to the "fifth stage". One hypothesis is that it will be a repetition of the previous four stages but assimilating the new technologies of minis/micros, word-processing and databases. Another hypothesis is that the effect of the new technologies will differ far more according to the organization involved than has happened in previous stages—probably because the new technologies will be far more integrated into the firm. His main hypothesis is, however, that this very integration of the "data resource function" into the firm will form a new stage that will go through the same "learning curves" as did the original assimilation of data processing technology:

> "Hypothesis: Stage V will be acceptance of the reorganization of data processing to appropriately align data resource management functions" (Nolan, 1977).

However, such projections for the "data resource function" are currently far from reality: practical database implementations are still basically straightforward information storage and retrieval systems; the more advanced experiments under the aegis of "artificial intelligence" have produced some dramatic indications of what is to come but are largely expensive showpieces that perform well only in the hands of their designers. Some of the extensions needed to make distributed, dynamic database systems a reality are fairly obvious and well-known, e.g. protection mechanisms, natural language access, and more efficient implementation. Others are not so obvious—problems of

information credibility, contradiction, vagueness, structural constraints and internal dynamics. This paper is concerned with some of these non-standard problems that are arising in attempts to make the logic of database systems more closely match that of human practical reasoning. Particular attention is paid to links with developments outside the conventional database literature since many of the problems encountered are truly fundamental and under consideration in other areas of study.

2. Some problems of database dynamics

This paper is concerned with the development of database systems that go beyond present relational databases in coping with the following problems of database dynamics:

(a) *Items in the database that are themselves relations*: e.g. implications imposed by semantic constraints, physical laws or judicial legislation. Such items allow the database to be extended, both positively and negatively, with inferred items that may, or may not, be present—however, the actual extension is wasteful in space, and often impossible in practice, so that "theorem-proving" techniques taking into account the relations must be used for what are apparently straightforward data retrieval or update, requests.

(b) *Action-initiating items in the database*: e.g. such that the entry of an item giving the level of a stock may initiate a re-order process when the value becomes low, or that requests for confidential information are aborted and security action taken if the enquirer has insufficient status.

(c) *Indeterminate or contradictory items in the database*: e.g. not just that an item is, or is not, in the database but that currently no information has been received on it, or contradictory information has been received. Clearly this is also related to the problem of the "credibility" of the information received.

(d) *Vague items in the database*: e.g. that someone's age is specified as not, for example, "25 years", but rather as "between 20 and 30 years", or "about 25 years", or "fairly young". All these are informative statements that would be useful in answering queries or making inferences dependent on the age, but they are a different form of data from that normally considered.

In the original context of databases as "filing systems" these requirements did not arise. Even in further developments of person-oriented interactive query languages it is possible to neglect them. However, as we move towards dynamic, multi-user, multi-centre database systems, such considerations assume great importance. Is this centre to accept updates from this remote source and give them equal credibility to those from its local controller? When this centre receives information which it passes on to another centre that promptly sends processed data back to the originator, what is the stability of the network (it is easy to invent cases that oscillate wildly!)? And so on—every problem of control and communication ever encountered elsewhere can be essentially reproduced in a database system—we are discussing structures that approach the ultimate generality of system theory and yet are very concrete.

That the problems mentioned are not highly artificial is best seen through an example conversation. Envision M as a manager and S as an information system and resource controller, eventually to be automated—the line numbers are for later reference:

1 M: We have a new employee starting tomorrow
2 S: Name?
3 M: Peter Grimes
4 S: Will he work in your division?
5 M: Yes
6 S: What section?
7 M: Either Sales or Distribution
8 S: Age?
9 M: Fairly young
10 S: Own car?
11 M: Don't know
12 S: Salary?
13 M: 15,000 for the first 3 months and then 17,500
14 S: Location?
15 M: Is 408 very full?
16 S: Yes
17 M: What else have we got?
18 S: 319 has room for one more
19 M: OK
20 S: I will order a desk and chair
21 M: How is my staffing now?
22 S: You have one vacancy in product planning
23 M: Productivity norms?
24 S: OK except for L division down 15%
25 M: Suppose I move Jane from L to G
26 S: All OK
27 M: No, that would be a demotion—note that demotions are now allowed
28 S: You cannot move Jane

FIG. 1. A sample dialogue.

I will discuss the natural language aspects of automating the S side of this conversation later. For the moment note the following points that arise naturally but create difficulties for a standard system:

(a) at line 7 the reply should be single-valued but actually gives two possible alternatives—we cannot treat this as "not information yet" because the data given is actually adequate to answer some queries, e.g. "list staff in Sales or Distribution"—however, we normally would not be able to store this reply or, in a relational database, would have the dual value flagged as an integrity violation;
(b) at line 9 the imprecision of the reply creates a rather worse problem through its vagueness—probably even a human S would have made some rather testy query about this reply—however, it is again an information-bearing reply that should not be ignored if it is the only information currently available—it would enable us to answer an enquiry about "employees not within five years of retirement" without any vagueness since a "fairly young" employee would fall into that class;
(c) line 11 is actually "no information yet"—note that this is quite distinct from "no car", e.g. in response to the query, "which employees do not have their own cars";
(d) line 13 is similar to line 7 in giving two values for a single-valued field but now involves a temporal trigger—it is not our uncertainty about the value but rather its own dynamics that leads to this reply—accepting this reply would involve storing a procedure that changes the salary after 3 months;
(e) line 15 involves imprecision again but this time in the query—note the essential terseness and simplicity of this query conveying exactly the content of the question to be answered—it is *not* the same as the query, "Is there room for one more person

in room 408?", the literal answer to which might be "yes" when the answer to the actual query was actually even stronger than "no", e.g. "most certainly not"!;

(f) at line 20 the assignment of the value 319 to the new employee's "location" field triggers off an activity on the part of S to order new furniture—such actions triggered by changes in the database are actually implicit in the way in which many systems are used currently, i.e. programs are run to initiate activities according to the state of the database—however, it can lead to interesting problems as the action initiated interacts itself with the state of the database;

(g) at line 25 a query is made which involves a counter-factual assertion, "what would happen if . . ."—this is a common requirement that is the basis of modelling and simulation facilities—in a multi-user dynamic data base system the data entry and rollback required may involve very complex processing, particularly if the entry itself triggers off other actions—in particular, if temporal data is not stored with every item, then requests for future or past data must be treated as counter-factuals, e.g. "what will our salary bill per month be in six months time?—what was it this time last year?"; at line 27 a constraint is added to the database specifying that certain types of change are not allowed—the counter-factual at line 25 is then noted to involve a constraint violation at line 28.

And so on—this dialogue is not at all unreasonable and surely resembles what users have been promised databases will do for them, i.e. allow them to maintain a straightforward record of the situation of their organization and interact with it in a simple and natural way. I have avoided the realms of fantasy: the domain of discourse is very strictly limited and S displays no great "intelligence". The next section gives a brief model of relational databases leading into the following sections concerning possible extensions to meet the additional requirements illustrated above.

3. Relational database operation

For the purposes of this paper Codd's relational model of a database is the most useful because of its basic simplicity and generality. Codd himself (1970, 1974) and Date (1976) have put the case for a relational model in user terms, and commercially available systems, such as Tymeshare's FOCUS (1976), demonstrate that the relational model is also practically realizable. The abstract form of the model is:

—the Cartesian product, P, of a set of a finite set of variable-valued fields, $\{F_i\}$, forms a *database state support*;
—any subset of P, $S_i \subset P$, is a potential *database state*;
—a family of subsets of P, $S \subset 2P$, that are potential states of an actual database is called a *database state space.*

These are the standard definitions phrased in a slightly different terminology in order to bring out the nature of a database as a state-determined machine, or *automaton,* a particularly important concept in studying database dynamics. Data entry can now be seen as a process of applying an input to the database automaton causing it to change its state. A database from which any item may be removed corresponds to a *reversible* automaton—the potential states of a database correspond to the *reachable* states of the automaton, and so on (Arbib, 1969).

The problems of database systems may be conveniently classified under three main headings: data retrieval, data entry/modification and system implementation. Considering each in turn:

(1) *Data retrieval* comes first, apparently out of logical order since data entry always precedes it, because it is simplest to consider the problems of accessing databases already established by some means—the problem of retrieval from a *static* database is well-defined and reasonably straightforward to "solve". The main sub-problems are:
   (1a) *inference* of the answers to well-defined queries—this may range from direct data access to complex "theorem-proving";
   (1b) *generation* of well-defined queries from user requests—this may range from a one-to-one translation of requests in a stylized language to the syntactic and semantic analysis of natural language;
   (1c) *protection* of data from unauthorized access—this may range from simple "password" checks to complex information type, value and processing checks.

(2) *Data entry and modification* are often best treated as distinct problems since a purely additive database from which information, once entered, may never be removed or modified comes somewhere between the static case and the fully dynamic one in its complexity and problems. The main sub-problems are:
   (2a) *inference* of the changes to be made in response to well-defined update requests—this may range from simple data storage to complex chains of data dependent modifications;
   (2b) *generation* of well-defined update requests from user requests—again this may range from trivial translation through to natural language "understanding";
   (2c) *integrity* preservation of the database information and structure—this may range from simple checks on authority of data source to complex determinations of the continued correctness of global predicates across the complete data base;
   (2d) *reversal* or "rollback" of modifications is generally required both to take care of error conditions, e.g. and also to allow counter-factual modifications to be made temporarily to the database, for example, for modelling possible changes that have not yet occurred.

(3) *Implementation* of database systems is still a key problem even for the current generation. Only in recent years have the storage costs of high-speed backing stores dropped to a level where it is possible to keep on line substantial amounts of data. To demand also that the data is not only accessible through its storage location but also through its structure and context places additional requirements on the speeds of such mass storage that can multiply the expense up to unrealistic levels. Thus, the efficient implementation of database systems is a major problem and any discussion which ignores this is somewhat unrealistic. However, for much of this paper I will not allow current implementation difficulties to rule out potentially useful techniques—the pace of hardware development and changes in system concept are too rapid for unreasonable pessimism about future implementations (Dolotta, Bernstein, Dickson, France, Rosenblatt, Smith & Steel, 1976). As a temporary developmental strategy this is sound—it is the approach which has lead to

fundamental progress in natural language systems in "artificial intelligence" research—systems which at first appeared as highly expensive research tools have now led to practically useful systems for natural language database retrieval at reasonable cost as will be illustrated in the next section.

4. Natural language interaction

The dialogue of Fig. 1 was in natural colloquial English and that in itself may be seen as a major problem. However, since the linguistic breakthroughs that came first with Weizenbaum's (1967) ELIZA that could maintain conversational niceties without "understanding", and then Winograd's (1972) SHRDLU that could capture much of the deep structure of English through its very "understanding" of a rather narrow domain, there has been a tremendous advance in the linguistic technologies enabling us to coat a core of basically algebraic intercourse between person and computer with the sugar of English language. Actually the significance of this goes far beyond the sweetening of the naïve user's pill—as system designers it is far easier for us to see the weaknesses of our systems when they appear as lack of power to interpret "obvious" English constructions than as algebraic/logical weaknesses. Indeed, since algebraicists and logicians have never come anywhere near expressing the range and variety of semantic structure and reasoning patterns exhibited by people, it would be surprising if an approach based on classical formal structures did not give a mis-match to what people actually require.

Petrick (1976) has given a useful survey of work on natural language based computer systems and some of the arguments for and against its use. The most dramatic demonstrations of the power of current techniques are more recent than his article, and it is worth describing briefly some of the capabilities of Hendrix's (1977) LIFER and Harris's (1977) ROBOT.

LIFER is implemented as a database access system in PDP10 Interlisp and used an augmented finite-state transition network (Woods, 1970; Norman & Rumelhart, 1975) to parse and semantically interpret questions in English. It has language specification functions that allow the transition networks to be constructed automatically from production rules readily comprehended by the specific system designer. These rules are readily modified and tested interactively allowing complex language constructs to be built and tested in a short time. Much of the semantics of the possible values in database fields is imbedded into the networks built by LIFER, e.g. "Gary" and "height" would be treated not just as "nouns" but as "person" and "attribute", respectively. LIFER is fast in operation, parsing, for example, "What is the length of the fastest American sub" in 2000 milliseconds (KL10 processor), and fairly small in size, requiring only 14K words extra over Interlisp itself. Hendrix notes that an extensive language definition for communicating with a fairly large database of 100 fields on 14 files with hundreds of records requires only an additional 33K words including some database access routines. It is particularly good at handling ellipsis where the user omits part of a sentence, e.g.:

1 M: What is the speed of the Kitty Hawk
2 S: Speed 35 knots
3 M: Of the Ethan Allen
4 S: Trying ellipsis: what is the speed of the Ethan Allen
Speed 30 knots

```
 5   M:  Displacement
 6   S:  Trying ellipsis: what is the displacement of the Ethan Allen
         Standard-displacement 6900 hundred-tons
 7   M:  Length of the fastest Soviet sub
 8   S:  Trying ellipsis: what is the length of the fastest Soviet sub
         Length 285 feet/speed 30 knots
 9   M:  Who onws the Kiev
10   S:  owns
         Country USSR
11   M:  Who owns the JFK
12   S:  Trying ellipsis: ellipsis failed
         The parser does not expect the word "JFK" to follow "who owns the"
         Options for the next word or meta-symbol are:
         (ship-name)
13   M:  Define JFK to be like Kennedy
14   S:  Country USA
```

FIG. 2. Sample dialogue with LIFER (Hendrix, 1977).

This sample shows that the terseness of my invented dialogue of Fig. 1 is fully realistic—computer-based natural language systems can emulate the cryptic style of colloquial English dialogue. At line 10 the Interlisp spelling corrector is used to rectify an error—this illustrates the way in which a standard module (Teitelman, 1975), once it exists, is used routinely by other systems to improve their utility. At line 12 the parse fails but note the excellent diagnostics provided (in technical rather than conversational form in this example) which project to the user what went wrong in readily comprehensible and rectifiable form. At line 13 the user gives an ELIZA-style definition which allows the parse now to be successfully completed.

ROBOT is another natural language enquiry system, this time on an IBM370/155 interfaced to the ADABAS database management system. It has been tested by routine use since January 1977 at Bigelow-Sanford Carpet Co. in connection with their customer and sales files (about 60,000 records). ROBOT is again based on an augmented transition network and Harris notes that it brings five types of information to bear on the translation process.

(1) *The merging of individual English words* is built-in for common words that are independent of the database, e.g. "is", "not", "greater", etc.
(2) *The syntactic structure of English* is represented by an augmented transition network in which each finite-state machine for parsing a syntactic structure is allowed to run, effectively in parallel. Thus ROBOT's parsings are non-deterministic at this level since it will produce all possible analyses.
(3) *Advice from the database administrator* is used to tailor the system to the expected linguistic forms of particular user communities, e.g. definitions of words likely to be used, default information likely to be implied, and navigational information on how the database is structured. Harris calls this "advice" because it can be explicitly overriden by the user in his enquiries.
(4) *The database itself* is used by ROBOT in translating a request to provide information regarding the use of individual words within it, and to help determine when interpretation on an enquiry makes sense with respect to the database.
(5) *The user himself* is the ultimate resort through a clarification dialogue if ROBOT cannot resolve ambiguity in any other way.

The following dialogue with ROBOT illustrates some of these features—the alternative lines given are examples of enquiries that would elicit the same reply:

```
1    M: Give me the names of all employees who have jobs working as secretaries in the city of
        Chicago
1'   M: Print the names of all the Chicago employees working as secretaries
1"   M: Who are the Chicago secretaries
1"'  M: Secretaries in Chicago
2    S: Print the name of any employee with city = Chicago and job = secretary
        The number of records to retrieve is
        Reinstedt
        Lawler
        Mooers
        .....
3    M: What is the family status of the area managers that live in New York
4    S: Request ambiguous. is it:
        Number 1
        Print the family status, state and city of any employee with state = NY and job = area manager
        Number 2
        Print the family status, state and city of any employee with city = New York and job = area
        manager
        Please type the number 1 or the number 2
5    M: 1
6    S: The number of records to retrieve is 120
        Married    NY    New York
        Married    NY    White Plains
        Divorced   NY    White Plains
        Married    NY    New York
        .....
7    M: Los Angeles area managers
8    S: Print the name of any employee with job = area manager and city = Los Angeles
        The number of records to retrieve is 14
        Atherton
        Holzbauer
        Needham
        .....
```

FIG. 3. Sample dialogue with ROBOT (Harris, 1977).

Line 1 of this dialogue is a request in a form where the syntax and semantics of the language make the decoding straightforward even if certain key words are replaced by nonsense words, i.e. the database itself need not be involved until a specific request for information has been formulated. However, at line 1"' the same request is formulated in such a way that knowledge of the structure of English is of little use—the form of the request can only be inferred by using the database itself to determine the nature of the words used. Then the inbuilt "advice" of the administrator must be used to infer the likely objectives of the request.

Harris (1977) and Pruitt (1977), the Data Processing Manager at Bigelow, report a very high degree of acceptance of ROBOT by ordinary users of the database.

> "The end user no longer has to think about files and fields in the sense that you and I do. Rather the user properly thinks about information, regardless of the source of that information or how it is stored."

Harris warns against extrapolating the ROBOT dialogue to the assumption that it can cope with arbitrary English conversation. However, within the framework of its own

domain of knowledge it does accept a wide range of natural language questions and is properly tolerant of syntactic "errors", ellipsis and so on, as would be any native speaker.

Returning to Fig. 1, I would wish to argue that the use of natural language involved is not beyond currently available computational linguistic technologies. The subtleties in the early part are just those that we are already aware of in programming far more formal dialogues (Gaines & Facey, 1975), e.g. at line 4 S does not ask "Sex?" but infers it from the name "Peter" (using a table look-up rather as in the spelling corrector)—however, since name → sex is not infallible the term "he" is introduced at line 4 to allow M to correct the inference if necessary, and so on—the use of currently available linguistic and dialogue programming techniques can cope with making the conversation in itself acceptable and attractive to the user. The real problems drawn out of Fig. 1 are instead ones of the basic "logic" of the database, rigid concepts of "truth", static views of the world, and so on. It is the actual structures within the database, the forms of information we can store, the patterns of inference available with it, etc., that need radical development. In the following sections I will give some more detailed examples of the problems and related developments.

5. Constraints, possible worlds and modal logics

It would be rare in any real database for the full potential range of database states defined in section 3 to correspond to the actually possible range, i.e. a database state space is a proper *sub-set* of its potential states. If we consider the database as an automaton whose inputs consist themselves of positive and negative sub-sets of P (i.e. items to be added to, and taken from, the database), three types of illegal transitions (subtractions) may be requested:

(1) additions (subtractions) to the state sub-set which lead to an impossible state sub-set such that no further additions (subtractions) can lead to a legal state;
(2) additions such that there exists a unique minimal sub-set of further additions (subtractions) leading to a legal state;
(3) additions (subtractions) such that there exist several incomparable minimal sub-sets of additions (subtractions) leading to a legal state.

The first case is such that for integrity the change must be rejected; the second case allows a unique "completion" of the change that preserves integrity; and the third case may be regarded either as offering a choice of "completion" or as necessitating rejection because of its indeterminacy.

Studies of database "normalization" have been primarily concerned with constraints on the database that lead to case 2, and with the implementation of the database (as a family of projections of the product set P) that automatically enforce the completion of requests. The use of strong and weak functional dependencies between sets of fields to factor the product into a family of projections in *fourth normal form* allows entry and update requests to be treated as single changes in projections that together generate a multiple change in the database state that is legal and preserves integrity (Fagin, 1977*a*). The normal forms are important because the database projections involved give a simple model of the database structure (as the sum of its projections)—Addis (1977) has pointed out how these projections serve as a "model of the organisation".

Clearly not all constraints on database updates can be expressed in terms of normalization, e.g. it may be that certain combinations of values in fields have been declared illegal for individual reasons that reflect no structural constraints. In the general case we can say that potential database states form a set of "possible worlds", a terminology dating back to Leibnitz but used in recent years to describe a key concept in modelling the semantics of modal logics (Snyder, 1971). What is particularly interesting from this point of view is that logicians commencing with intensional notions of modal concepts such as possibility and necessity, permission and obligation, etc., have arrived in the last decade at semantic systems that are basically extensional using structures of "state-descriptions" that are formally identical to those of database theory. This has enabled many of the fundamendal logical difficulties of formal reasoning with modal concepts to be overcome, so that we now have well-grounded theories for a wide variety of modalities and it is feasible to attempt to formalize many forms of practical reasoning. However, modal logics can also be viewed as a form of constraint analysis of possible worlds, i.e. is this state transition *permitted* (deontic modality), *possible* (alethic modality), *credible* (epistemic modality), etc., and hence are very directly related to the integrity and normalization of databases.

An illustration of this relationship comes, for example, from a comparison of Lewis's (1973) analysis of counter-factual conditionals in philosophical logic and Todd's (1977) analysis of automatic constraint maintenance in databases. Lewis develops a model of the relationships of possible worlds in terms of a partial order of similarity—Todd defines just such a partial order over database states in order to determine the minimal completion of updates necessary to preserve integrity constraints as defined in (2) above. Lewis is concerned to analyse such statements as:

"If kangaroos had no tails, they would topple over"

and notes the problems of considering a possible world satisfying the first clause:

> "We might think it best to confine our attention to worlds where kangaroos have no tails and everything else is as it actually is; but there are no such worlds. Are we to suppose that kangaroos have no tails but their tracks in the sand are as they actually are? Then we shall have to suppose that these tracks are produced in a way quite different from the actual way."

If one replaces his example with:

"If we ship no orders for a week, our cash flow will deteriorate"

the relevance to commercial databases may be more obvious! There are such close links, formal and informal, between the logicians analysis of possible world models for modal logics and current requirements for the constraint analysis of database systems that a mutual interchange of ideas and problems would seem to be very fruitful. We have already indicated some of the relationships in more detail for the Graham & Denning (1972) model of protection structures (Kohout & Gaines, 1976).

Even when the extension of the database system to counter-factual analysis is not required, the concept of possible worlds can be important. The situation at line 7 of the dialogue in Fig. 1 where two possible values are given to a single-valued field is a simpler requirement but leads to similar problems. We cannot put both alternatives in the database without violating the (presumed) constraint that an employee is assigned to one section only. Neither can we simply mark the entries as "possible" since this does not carry the information that at least one of the possibilities must occur and that they

cannot both occur together. That is, in modal logic terms, we have to account not only for:

Possible(Sales) **AND Possible**(Distribution)

but also:

Necessary(Sales **OR** Distribution) and
Not Possible(Sales **AND** Distribution)

This is a situation similar to that in automata theory where we might know that an automaton is possibly in each of several states, but with the over-riding constraint that it must be in one, and not more than one, state. We have shown (Gaines & Kohout, 1976) that this type of situation may be represented by a weakened form of probability logic (Rescher, 1969) in which the value 0 means **impossible**, 1 means **necessary**, and intermediate values mean **possible**. The constraint that one and only one of the possibilities must occur is then equivalent to the sum of assigned "probabilities" being 1. In database terms this corresponds to mapping the characteristic function of the relation P not into the two points {0, 1} but into the complete interval [0, 1].

6. Paradoxes, knowledge and action

The close relationships known to exist (Date, 1976; Fagin, 1977*b*) between databases and systems of logic prompt one to enquire whether the paradoxes of logic may also be found in database operation. Tokuda (1977) has recently pointed out that the axiom of comprehension in naïve set theory that leads to Russell's paradox is implicitly assumed in some of the computing science literature, i.e. we cannot assume that every predicate leads to a well-defined set. Russell's "barber" paradox has a direct representation in terms of a database of information on the person who shaves each individual in a village subject to the constraints that:

(1) if someone does not shave himself he is shaved by the barber;
(2) if someone shaves himself he is not shaved by the barber.

Suppose that items in this database are pairs, (X, Y), representing the relation, "X shaves Y", and let the barber be B. Consider the pair (B, B)—does it belong to the database or not? Starting with no (B, B) in the database and enforcing constraint (1) we have to add (B, B) to preserve integrity. Then applying constraint (2) we find that we have to remove (B, B) to preserve integrity. Thus the net effect of the two constraints is that the state of the database will oscillate between two positions, and an enquirer will sometimes find that "B shaves B", sometimes not. Pinkava (1977) has shown that many classical paradoxes of logic may be represented as oscillations in networks. Clearly, the imposition of a family of constraints can lead to very much more complex autonomous behaviour within the database, basically deriving from the known *inconsistency* of naïve set theory (Frankel, Bar-Hillel & Levy, 1973).

An oscillation in itself may be seen as a local problem that reflects the ambiguous status of certain items of information, i.e. (B, B) should be both in and out of the database (e.g. one resolution of the Russell paradox is to assign this data the truth-value 1/2 (Gaines, 1976*b*)). However, global problems destroying the integrity of the entire database can arise basically through similar phenomena to those encountered with

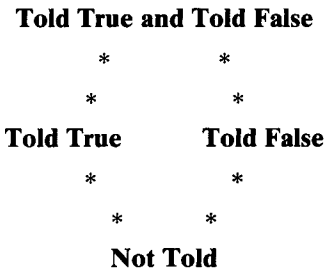
asynchronous interacting processes, e.g. picking up a supposedly static value not realizing that it is being changed whilst it is being used. If P is an oscillating proposition and Q is *any* other proposition then we can infer that Q is true by a sequence such as:

> Examine P—TRUE—hence infer that (P OR Q) is TRUE and put away for later use.
>
> Examine P again—FALSE—hence from (P OR Q) is TRUE infer that Q is TRUE.

Using information within the database itself to determine the form of an update may be viewed as the imposition of an integrity constraint at data entry, and similar phenomena can arise. In general, if we allow entries to a database to be determined, or partially determined, by entries already there, then the entry of one item of information may so change the database that the item should no longer be entered.

Even simpler examples of conflict between the state of a database and data to be entered may arise if we distinguish between entries and updates and validate new entries against the data already entered. Belnap (1976) considers the problem of a real-world database and suggests that the two situations of no information being entered on a particular item, and contradictory information being entered, cannot be avoided. He notes that one cannot represent these situations adequately in a 2-valued logic that only mirrors the truth or falisity of statements, and takes no account of the epistemic state of the database, i.e. one must distinguish "John has a car" is **true** or **false**, from whether "John has a car" is **known** or **unknown**, and one must allow for both "truth" and "falsity" to be known together without the entire logical system for inferences about the database breaking down.

Belnap proposes a 4-valued logic for databases with the values forming the lattice:



and demonstrates that a simple and consistent logical system may be built on this. In this system the "barber" paradox causes no problems of oscillation because the contradiction involved is acceptable, i.e. (B, B) will be **Told True and Told False** which seems a reasonable resolution of that problem!

Clearly, a move to multi-valued logics does allow more information to be carried about the status of information in a database. However, whilst even a 3-valued logic is known to be adequate to deal with Russell's paradox (Skolem, 1960), there are higher-order paradoxes (corresponding to more complex systems of constraints in this context), that are known to be irresolvable by any finite-valued logic (Chang, 1964; Fenstad, 1964; Maydole, 1975). It is also known that the modelling of modalities, as discussed in the previous section, is not possible in any finite-valued logic (Dugundji, 1940) and even that paradoxes can occur in naïve set theory based on the infinite-valued probability logic (Maydole, 1975) that represents the modal logic S5 (Rescher, 1963). In practice,

the significance of a requirement for "infinite-valued" logics is not so unrealistic as it might seem—really it means that the number of values in our logic must be extensible to cope with situations of increasing complexity. That is a common requirement for many fields in a conventional database—in a dynamic real-world situation sooner or later we may run out of bits to represent a possible range of values.

The requirement for more than the 4 values suggested by Belnap will also arise in a real system in other ways—if we have **Told True and Told False**, our next question will probably be "Who told me true and who told me false?"! Indeed, even without contradiction, if we have "**Told True**" for a second time this may well increase the *credibility* of the information actually being true—Giles (1976) has pointed out that conjunction and disjunction are not necessarily idempotent in practical reasoning—we may emphasize the truth of a statement by repeating it. This phenomenon would occur naturally if the database were treated as an *inductive* inference system (Gaines, 1977) in which incoming data was treated as evidence on which to establish truth rather than statements of truth in themselves. It also occurs in these systems of vague reasoning discussed in the next section where all statements are treated as being imprecise so that the precision of knowledge may increase as related information is received. In general, for large multi-user, multi-centric databases some means of carrying information about the epistemic status of the database, its "state of knowledge", seems essential.

A number of systems for the representation and acquisition of "expert" knowledge have used multivalued logics as an intrinsic feature of their computations. Sridharan & Schmidt (1977) in "Believer" use a 3-valued logic with 3 of Belnap's values: **True/False/Unknown**. Isner (1975) in his "understanding" system and Shortliffe (1976) in "MYCIN" both discuss and use many-valued systems. Michalski (1977) has developed a range of quantified variable-valued logics that have been used as the basis of both inductive knowledge acquisition systems and in expert knowledge representation.

The actual imposition of integrity constraints can clearly take place as part of the update routine or as a separate set of processes "woken up" when certain preconditions are met within the database. In a complex database system with many possible update paths where integrity constraints may themselves be dynamic variables (as at line 27 of Fig. 1) the use of *data-driven* processes, "demons" or "actors" (Hewitt & Baker, 1977) is attractive. Hendrix (1973) has shown how dynamic processes may be simulated by such data-driven processes and other workers have given a variety of detailed examples of the modelling of physical systems (Reiger & Grinberg, 1976; Lowrance & Friedman, 1977), e.g. the "integrity constraint" that one cannot move through a brick wall is modelled by a process that wakes up when an object comes into contact with the wall and cancels further motion of that object. Smith & Hewitt have suggested that many of the goal-directed activities of the PLANNER system used by Winograd (1972) in implementing SHRDLU may be achieved through simple mechanisms based on data-driven processes, and have proposed a language Plasma (Smith & Hewitt, 1975) based on this. Even in the context of far simpler problems of information access protection (Conway, Maxwell & Morgan, 1972) and languages providing it such as EL1 (Wegbreit, 1974), the detection of access violations is really only practically feasible using hardware mechanisms such as tagged (Feustel, 1973) or descriptor-based (Gaines, Facey, Williamson & Maine, 1974) machines allowing the type of mechanisms that Zelkowitz (1971) has proposed for the IBM360.

It is interesting to note again that philosophical logicians have also treated the problems that arise in reasoning systems determining actions including the types of conflict that lead to oscillations (Rescher, 1967).

7. Imprecision and vagueness

The initial applications of computers were in the exact sciences and comparable commercial areas such as accountancy. Database theory and practice in developing from this, even though it allows for non-numerical data, still requires it to be precise and well-defined. As applications move out of the realms of the accountant and into the less quantitative areas of the firm so also does a requirement for high precision in specifying retrievals and updates of a database become increasingly unnatural. It is possible to argue that precision in itself is always a virtue—a reply to a request for a delivery date that says "soon" would be rated less satisfactory than one that says "in 3 days time"—the latter is more "businesslike".

However, unwarranted precision can itself be highly misleading since actions may be taken based on it—"we will deliver 7 parcels each weighing 15·2 kilograms at the rear entrance of building 6A on 15th February at 0.03 p.m.", "we will deliver some heavy equipment to your site Saturday evening", and "see you with the goods over the weekend", may each refer to the same event but are clearly not interchangeable, i.e. each conveys an exact meaning that (presumably) properly represents what is to occur. If we prefer the precision of the first statement it is not for its own sake but because the tighter tolerances it implies on the actual situation allow us to plan ahead with greater accuracy and less use of resources. However, if the third statement really represents all that can be said it would be ridiculous to replace it with either of the previous ones. It would be equally ridiculous to say nothing, however, since even the least precise of the three statements does provide a basis for planning and action. A key aspect of executive action is planning under uncertainty and normal language provides a means for imprecision to be clearly and exactly expressed.

In retrieving information from a database the requirement for artificial precision is at least irritating and at worst highly misleading, e.g. the request, "list the young salesmen who have a good selling record for household goods in the north of England", is perfectly comprehensible to a person. Translating it into, "list salesmen under 25 years old who have sold more than £20,000 of goods in the categories ... to shops in the regions ...", generates unnecessary work and makes no allowances for the whole spectrum of trade-offs possible, i.e. it will not list the chap of 26 who has made a real killing, or the one of 19 who sold £19,000, etc. The second request is more precise but the first represents far more accurately the actual *meaning* of the retrieval required.

Zadeh (1965, 1976, 1977, 1978*a*,*b*) over many years has argued the need for systems engineering to be based on "linguistic reasoning" that accurately reflects the imprecision of most real-life situations, and not the artificially quantitative approaches that form the stereotype in modern "science". He has developed a complete system of *fuzzy logic* that allows imprecise, but practically valid, arguments in everyday reasoning to be translated into computable terms. Over the last decade there have been widespread developments in the foundations of such fuzzy reasoning (Goguen, 1974; Gaines, 1976*b*) and its applications to problems of control, pattern recognition, linguistics, etc. (Gaines & Kohout, 1977). Recently Zadeh (1978*b*) has developed a language PRUF

which he terms "a meaning representation language for natural languages" that is well-suited to database applications.

The foundation for PRUF is Zadeh's "fuzzy set theory" in which the characteristic function of a set is extended to range throughout the interval [0, 1] instead of taking values only at its end-points. Zadeh terms the value of the characteristic function of a given element of a sub-set its "degree of membership" to that sub-set, and represents imprecise terms as "possibility distributions" of degree of membership to the possible values, i.e. the statement, "Louie is on a high salary" assigns a possibility valuation to each level of salary that Louie might have. Thus such an imprecise statement as an entry to a database can be regarded as corresponding to entering in the database a "fuzzy relation" consisting of a number of possible relations each with a defined degree of membership to being true. Zadeh (1977, 1978*a*,*b*) develops in detail an inference system for answering queries about such a fuzzy database.

Clearly, implementing such a fuzzy database by actually storing all the relations generated by an imprecise statement together with their degrees of membership would lead to space problems. It is more realistic to consider storing the original linguistic statement and generating the implied possibility distributions only as required. It is also possible to develop higher level patterns of vague reasoning at the linguistic level that are implied by the possibility distributions but are computationally independent of them—such an approach has proved fruitful in the analysis of fuzzy linguistic controllers (Rutherford, 1979).

8. Summary—the logics of databases

Logics of possibility, induction, imprecision, action, etc., go beyond the classical predicate calculus and are not representable within it. Attempts to do so lead to paradoxes, false inferences and insufficiently rich semantics to express what is required. The common pattern of development for the resolution of these different problems is to go over to *multi-valued* logics allowing a range of "truth-values" in the interval [0, 1] instead of just its two end-points. In terms of Codd's relational model outlined in section 3, this corresponds in the simplest cases to taking the database states not as standard sub-sets of the Cartesian product, P, i.e. mappings to {0, 1}, but instead as mappings from P to the entire interval [0, 1].

However, a number of problems arise:

(1) The multi-valued logics required have to be infinite-valued. This, at first sight, is absurd in computational terms. However, in practice it turns out that it is the extensibility of the truth-values (formally that they be *dense* in [0, 1]) that is required. We wish to be able to place a new truth-value, y, between two old ones, x and z, such that $x < y < z$. Clearly, if on the occasion we have to do this it turns out that the computer word-length is such that there are no values left between x and z then the inference scheme will begin to break down. This is just the way in which the manipulation of floating point reals breaks down and is nothing new computationally—indeed it suggests a role for the numerical analyst in areas of theorem-proving that have so far been outside his province!

(2) Actually storing relationships in extensive form is unrealistic when we no longer have available the computational trick of a two-valued world where items mapping

to 1 may be stored and items mapping to 0 left out. However, in many instances it seems far more natural to store the generator of a relation than the extension of it—a person would need a rather large cortex to hold all integers, for example! In practice we have to store some isolated "facts" and much procedural "knowledge" and use the underlying extensional calculus to make inferences combining both.

(3) The logical rules that apply to the multi-dimensional logics generated for different purposes can be very different in their connectives, e.g. the natural logic of possibility is a weakened probability logic whereas that of imprecision and vagueness is Łukasiewicz logic (Gaines, 1975, 1976*b*). The connectives of one logic are arithmetic operations of addition and subtraction, whereas those of the other are comparisons leading to minimum and maximum, and, even though these may be shown to be related (Gaines, 1976*a*), they appear quite different. It turns out that in situations where possibility, credibility, imprecision, etc., combine, we require a set of relevant multi-valued logics operating on a *possibility vector*—Gaines (1975) develops such a system for possibility, probability and eventuality.

These considerations leave us with a range of difficult problems to be solved before multi-valued logics may be applied to real database systems. However, only a decade has passed since logicians finally came to grips with the decision procedures of standard modal logics studied since the time of Aristotle. Computational techniques to deal with a range of modal logics are now available, although the problems of mixed modalities are still largely open. Some workers such as Creswell (1973) and Montague (1974) have developed large-scale systems for the logical analysis of natural language. The database as an abstract entity is very closely related to the possible worlds models of modal logics, and as a practical system it is very closely related to the linguistic structures that these logics formalize. I hope this paper will encourage further research into the common problems of database systems, linguistics and philosophical logic.

9. Conclusions

This paper has raised many problems of database dynamics and logic for the next generation of systems and briefly outlines some possible approaches to their solution. This is a fair reflection of the current state-of-the-art where the effort to implement and operate conventional database systems has precluded all but a few explorations of some of the issues raised here. However, the technologies for database support are developing very rapidly indeed and, as has already happened to a large extent with high-level languages, the hardware/software available may soon outstrip current user requirements. We need to think ahead and determine what limitations of current database concepts will prove to be severe restrictions in future applications without paying too much regard to current implementation problems.

I have suggested that the sheer mechanics of natural language communication with databases may prove to be far less of a problem than we have tended to assume—developments such as LIFER and ROBOT are highly encouraging. However, the "world-view" incorporated in the databases driving these systems is highly restricted not so much in the quantity of information stored but in its quality, richness and fitness, compared with that used in comparable human reasoning within an organization. The various sections of this paper illustrate some of the difficulties that may arise and survey

some of the techniques currently being investigated to overcome them. As with many of the problems of computing science, and system engineering in general, the solutions themselves seem to present far less difficulty than does the appropriate problem formulation. If we can define what we are trying to achieve then the problem is 90% solved. I hope this paper will encourage wider discussion of the features required in the next generation of databases.

I am grateful for access to related material and discussions on topics treated in this paper with Robin Giles, Joe Goguen, Susan Haack, Larry Harris, Ladislav Kohout, Ebrahim Mamdani, Ryszard Michalski and Lotfi Zadeh.

References

ADDIS, T. R. (1977). An introduction to relational analysis for data-base design. Paper presented at National Conference of Operational Research Society (UG.21), Sept., International Computers Ltd., Stevenage, Herts., U.K.

ARBIB, M. A. (1969). *Theories of Abstract Automata.* Englewood Cliffs, N.J.: Prentice-Hall.

BELNAP, N. D. (1976). How a computer should think. In RYLE, G., Ed., *Contemporary Aspects of Philosophy.* Stockfield: Oriel Press, pp. 30–56.

CHANS, C. C. (1964). Infinite valued logic as a basis for set theory. In BAR-HILLEL, Y., Ed., *Proceedings of 1974 International Congress for Logic, Methodology and Philosophy of Science.* Amsterdam: North-Holland, pp. 93–100.

CODD, E. F. (1970). A relational model of data for large shared data banks. *Communications of the ACM,* **13** (6), 377–387.

CODD, E. F. (1974). The relational approach to data base management: an overview. *IEEE 1974 Texas Conference on Computing Systems* 74-CH0895, 6.1.1–6.1.5.

CONWAY, R. W., MAXWELL, W. L. & MORGAN, H. L. (1972). On the implementation of security measures in information systems. *Communications of the ACM,* **15** (4), 211–220.

CRESSWELL, M. J. (1973). *Logics and Languages.* London: Methuen.

DATE, C. J. (1976). *An Introduction to Database Systems.* Reading, Mass.: Addison-Wesley.

DE BONO, E. (1979). *Future Positive.* London: Maurice Temple Smith.

DOLOTTA, T. A., BERNSTEIN, M. I., DICKSON, R. S., FRANCE, N. A., ROSENBLATT, B. A., SMITH, D. M. & STEEL, T. B. (1976). *Data Processing in 1980–1985.* New York: John Wiley.

DUSUNDJI, J. (1940). Note on a property of matrices for Lewis and Langford's calculi of propositions. *Journal of Symbolic Logic,* **5**, 150–151.

FAGIN, R. (1977*a*). Multivalued dependencies and a new normal form for relational databases. *ACM Transactions on Database Systems,* **2** (3), 262–278.

FAGIN, R. (1977*b*). Functional dependencies in a relational database and propositional logic. *IBM Journal of Research and Development,* **21** (6),

FENSTAD, J. E. (1964). On the consistency of the axiom of comprehension in Łukasiewicz infinite valued logic. *Mathematica Scandinavica,* **146**, 65–74.

FEUSTEL, E. A. (1973). On the advantages of a tagged architecture. *IEEE Transactions on Computers,* **C-22**(7), 644.

FOCUS (1976). *Focus Query Language,* California, U.S.A.: Tymshare Inc., June.

FRAENKEL, A. A., BAR-HILLEL, Y. & LEVY, A. (1973). *Foundations of Set Theory.* Amsterdam: North-Holland.

GAINES, B. R. (1976*a*). Fuzzy reasoning and the logics of uncertainty. *Proceedings of the 6th International Symposium on Multiple-Valued Logic 76CH1111–4C,* May. Utah State University, Logan, Utah, U.S.A., pp. 179–188.

GAINES, B. R. (1976*b*). Foundations of fuzzy reasoning. *International Journal of Man–Machine Studies,* **8** (6), 623–668.

GAINES, B. R. (1977). System identification, approximation and complexity. *International Journal of General Systems,* **3** (3), 145–174.

GAINES, B. R. (1978). Computers in world three. *Proceedings of the International Conference on Cybernetics and Society IEEE-78CH-1306-0-SMC*, Nov. Tokyo, Japan, pp. 1515–1521.

GAINES, B. R. & FACEY, P. U. (1975). Some experience in interactive system development and application. *Proceedings of the IEEE*, **63**, 155–169.

GAINES, B. R., FACEY, P. U., WILLIAMSON, F. K. & MAINE, J. A. (1974). Design objectives for a descriptor-organized minicomputer. *Proceedings of the European Computing Congress, EUROCOMP 74*, May, pp. 29–45.

GAINES, B. R. & KOHOUT, L. J. (1976). The logic of automata. *International Journal of General Systems*, **2** (4), 191–208.

GAINES, B. R. & KOHOUT, L. J. (1977). The fuzzy decade: a bibliography of fuzzy systems and closely related topics. *International Journal of Man–Machine Studies*, **9** (1), 1–68.

GILES, R. (1976). Łukasiewicz logic and fuzzy set theory. *International Journal of Man–Machine Studies*, **8** (3), 313–327.

GOGUEN, J. A. (1974). Concept representation in natural and artificial languages: axioms, extensions and applications for fuzzy sets. *International Journal of Man–Machine Studies*, **6** (5), 513–561.

GRAHAM, G. S. & DENNING, P. J. (1972). Protection principles and practice. *AFIPS Spring Joint Computer Conference 40*. New Jersey: AFIPS, pp. 417–429.

HARRIS, L. R. (1977). User oriented data base query with the robot natural language query system. *International Journal of Man–Machine Studies*, **9** (6), 697–713.

HENDRIX, G. G. (1973). Modelling simultaneous actions and continuous processes. *Artificial Intelligence*, **4** (3), 145–180.

HENDRIX, G. G. (1977). LIFER: a natural language interface facility. *Proceedings of the Second Berkeley Workshop on Distributed Data Management and Computer Networks TID-4500-R65*, May, pp. 196–201.

HEWITT, C. & BAKER, H. (1977). Laws for communicating parallel processes. In GILCHRIST, B., Ed., *Information Processing 77*. Amsterdam: IFIP, North-Holland, pp. 987–992.

KOHOUT, L. J. & GAINES, B. R. (1976). Protection as a general system problem. *International Journal of General Systems*, **3** (1), 3–23.

ISNER, D. (1975). Understanding "understanding" through representation and reasoning. *Ph.D. thesis* (University Microfilms 75-21,758). University of Pittsburgh, U.S.A.

LEWIS, D. (1973). *Counter-factuals*. Oxford: Basil Blackwell.

LICKLIDER, J. C. R. (1968). Man–computer symbiosis. In ORR, W. D., Ed., *Conversational Computers*. New York: John Wiley.

LOWRANCE, J. D. & FRIEDMAN, D. P. (1977). Hendrix's model for simultaneous actions and continuous processes: an introduction and implementation. *International Journal of Man–Machine Studies*, **9**, (5), 537–581.

MAYDOLE, R. E. (1975). Paradoxes and many-valued set theory. *Journal of Philosophical Logic*, **4**, 269–291.

MICHALSKI, R. S. (1977). Variable–valued logic and its application to pattern recognition and machine learnings. In *Multiple-Valued Logic and Computer Science*. Amsterdam: North-Holland.

MONTAGUE, R. (1974). *Formal Philosophy*. New Haven: Yale University Press.

NOLAN, R. L. (1973*a*). Computer data bases: the future is now. *Harvard Business Review*, 98–114.

NOLAN, R. L. (1973*b*). Managing the computer resource: a stage hypothesis. *Communications of the ACM*, **16** (7), 399–405.

NOLAN, R. L. (1977). Thoughts about the fifth stage.

NORMAN, D. A. & RUMELHART, D. E. (Eds) (1975). *Explorations in Cognition*. San Francisco: W. H. Freeman.

PETRICK, S. R. (1976). On natural language based computer systems. *IBM Journal of Research and Development*, **20** (4), 314–325.

PINKAVA, V. (1977). On the nature of some logical paradoxes. *International Journal of Man–Machine Studies*, **9** (4), 383–398.

POPPER, K. R. (1976). *Unended Quest*. London: Fontana/Collins.

PRUITT, J. (1977). A user's experience with ROBOT. *Proceedings of the Fourth Annual ADABAS Meeting*, April.

RESCHER, N. (1963). A probabilistic approach to modal logic. *Acta Philosophica Fennica*, **16**, 215–226.

RESCHER, N. (Ed.) (1967). *The Logic of Decision and Action.* Pittsburgh: Pennsylvania University of Pittsburgh Press.

RESCHER, N. (1969). *Many-Valued Logic.* New York: McGraw-Hill.

RIESER, C. & GRINBERG, M. (1976). The causal representation and simulation of physical mechanisms. *Computer Science Technical Report Series TR-495*, November. University of Maryland, Virginia, U.S.A.

RUTHERFORD, D. A. (1979). Theoretical and linguistic aspects of the fuzzy logic controller. *Automatica*, to appear.

SHORTLIFFE, E. H. (1976). *Computer-Based Medical Consultations: MYCIN.* Elseveier Computer Science Library (2). New York: Elsevier.

SIBLEY, E. H. (1976). The development of database technology. *ACM Computing Surveys*, **8** (1), 1–5.

SKOLEM, TH. (1960). A set theory based on a certain 3-valued logic. *Mathematica Scandinavia*, **8**, 127–136.

SMITH, B. & HEWITT, C. (1975). *A Plasma Primer.* Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, Mass., U.S.A., May.

SNYDER, D. P. (1971). *Modal Logic.* New York: Van Nostrand Co.

SRIDHARAN, N. S. & SCHMIDT, C. F. (1977). Knowledge-directed inference in believer. *Departments of Computer Science and Psychology CBM-TR-75*, January. Rutgers University, New Brunswick, U.S.A.

TEITELMAN, W. (1975). *INTERLISP Reference Manual.* Xerox Palo Alto Research Center, California, U.S.A.

TODD, S. (1977). Automatic constraint maintenance defined relations. In GILCHRIST, B., Ed., *Information Processing 77.* Amsterdam: IFIP, North-Holland, pp. 145–148.

TOKUDA, T. (1977). A question of style in future computing literature. *SIGART News*, **9** (2), 6–7.

WESBREIT, B. (1974). The treatment of data types in EL1. *Communications of the ACM*, **17** (5), 251–264.

WEIZENBAUM, J. (1967). Contextual understanding by computers. *Communications of the ACM*, **10** (8), 474–480.

WINOGRAD, T. (1972). *Understanding Natural Language.* Edinburgh University Press.

WOODS, W. A. (1970). Transition network grammars for natural language analysis. *Communications of the ACM*, **13** (10), 591–606.

ZADEH, L. A. (1965). Fuzzy sets. *Information and Control*, **8**, 338–353.

ZADEH, L. A. (1976). A fuzzy-algorithmic approach to the definition of complex or imprecise concepts. *International Journal of Man–Machine Studies*, **8**, 249–291.

ZADEH, L. A. (1977). A theory of approximate reasoning, *Electronics Research Laboratory Memorandum M77/58.* University of California, Berkeley, California, U.S.A.

ZADEH, L. A. (1978*a*). Fuzzy sets as a basis for a theory of possibility. *Fuzzy Sets and Systems*, **1** (1), 3–28.

ZADEH, L. A. (1978*b*). PRUF—a meaning representation language for natural languages. *International Journal of Man–Machine Studies*, **10** (4), 295–460.

ZELKOWITZ, M. (1971). Interupt driven programming. *Communications of the ACM*, **14** (6), 417–418.