TOWARDS A DATA MODEL
FOR ARTIFICIAL INTELLIGENCE APPLICATIONS.

Sergei Nirenburg, Colgate University

Chagit Attiya, Hebrew University

Abstract. Data models used in database management have not been built with AI applications in mind. The entities and their relationships in an AI environment transcend in complexity the data semantics of most other databases, so that the expressive power of the "usual" data models becomes insufficient. In AI community databases are viewed as a possible application area ("database front ends") but in AI research itself the databases used tend to be ad hoc and are not specified in terms of data models and DBMS based on such. The data model suggested below is a step towards bridging the gap between database theory and AI databases.

Introduction

The complexity of problems encountered by AI researchers designing both the knowledge-based expert systems and theoretical models of cognitive entities is notoriously high. The overall task involves many components, of which the most widely studied are knowledge representation languages, various "inference engines" (models of informal reasoning), automatic deduction, models of planning and cognitive processes, parsers and grammars. Although it is currently recognized that databases are integral parts of practically any AI system, and although a majority of AI systems employ the database concept, it is a fact that most such databases, however ingeniously built, are inherently ad hoc and, more importantly, do not, as a rule, reflect the developments in the theory of database management. A very good example is the approach described in the influential book by Charniak, McDermott and Riesbeck (1980). These authors devote much space to describing the AI databases (predicate calculus-based, slot-and-filler ones, etc.), mostly in terms of their ability to support ADD, ERASE and RETRIEVE operations, but they conclude by warning the potential user of the database approach in AI research about the costs of using a central database. They list two main disadvantages of the central DBMS:

1. It is inefficient to have to use a common data-structuring convention and transact all business through the manager. If a module [of an AI system] wishes to record the single fact that it is a leap year, it takes less time and storage to have a Boolean variable Leap-Year set to T than to keep

(= (CARDINALITY (DAYS THIS-YEAR)) 366)

... in the database...

2. If your database reaches a size where secondary storage is needed, then you have entered TERRA INCognita. No one knows how efficient a large random-access database can be" (p.228). There is no mention of the data models in the book and only a hint at the very important notion of data independence. In our opinion, any AI system being a modular and multi-layered one, the problem of "keeping an eye" on all the components at the same time, that is of system architecture is as much a problem of efficiency as the that of search and retrieval. It is mainly because of the use of the "blackboard" database manager that the HEARSAY systems of speech recognition (cf. Erman et al., 1980) have become so well-known.

It can be held that the responsibility for building a data model, database management system and a physical database design guidelines for AI is a responsibility of the workers in the database field. However, AI as an application for DBMS design has not yet caught on as a favorite topic among the database people. Note that the opposite problem: databases (especially "natural language front ends" to database systems) as an application for AI has attracted attention of AI researchers (cf., e.g. Waltz, 1977) But this work does not have much in common with our task, in which AI is the target domain and database management, the source.
As there are no scientific descriptions independent of some kind of theory, so there is no AI system that does not correspond to a certain model of the real world. Since AI systems are specifically designed to perform tasks involving cognition, this world model becomes a model of a cognitive system. Such a model may be intuitive and far from being well delineated. In the most abstract form, a cognitive system includes the following modules: input (perception), memory management (thinking processes) and output (performance). For a detailed discussion of a cognitive system structure cf. Norman (1980). Concrete AI applications (such as expert systems and natural language systems) generally do not concern themselves with this level of description, choosing instead to represent the subworld of the subject domain: spectrograms for DENDRAL (Buchanan and Feigenbaum, 1978), visiting restaurants for SAM (cf. e.g. Cullingford, 1981). The complexity of the research warrants this limitation but leads to the situation in which a number of research teams, though sharing the basic philosophy, tend to produce different working systems, based on separately developed representations and languages for their manipulation. Thus, the frame-based systems of E. Charniak’s group at Brown University (FRAIL, BRUI) and the systems of R. Schank, R. Abelson, C. Riesbeck and others at Yale (MARGIE, ELI, SAM, PAM, etc.), using the notions of scripts, plans, goals and MOPS, seem to share the essential approach to the study of cognitive models but develop different representations and processing systems.

Schematically, this state of affairs in AI data management is depicted in Fig. 1.

The diagram in Figure 1 is very schematic. In real AI systems the distinction between the programming language and the representation language may be blurred (cf. PLANNER, e.g. Hewitt, 1972) or CONNIVER (e.g. McDermott and Sussman, 1972). The dotted link between the “cognitive model” block and the application means that this link is not always recognized. Some people, notably Bobrow and Winograd, in their KRL paper (1977), draw a clear distinction between the data models and representation languages. Still, this “architecture” underscores the fact that data management in AI systems has not yet enjoyed the attention it deserves.

We would like to suggest a centralized architecture for the AI databases. Note that we do not suggest that the content of the cognitive system model must be the same for all the AI workers. We would like to suggest a technical data model which can support a variety of specific database instances. The proposed architecture for a system based on such an approach is presented in Fig. 2.

Over the course of the last decade the original conception of computer science as the field studying algorithms and programming languages has somewhat changed to reflect the new emphasis on data structures and databases (cf. Horowitz & Sahni, 1978, pp. 4-5, for an alternative definition of the field). In AI research, however, it is still customary to speak about AI programming.

---

<table>
<thead>
<tr>
<th>Cognitive system model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application 1 ..........</td>
</tr>
<tr>
<td>V</td>
</tr>
<tr>
<td>AI system control</td>
</tr>
<tr>
<td>module</td>
</tr>
<tr>
<td>V</td>
</tr>
<tr>
<td>[Programming]</td>
</tr>
<tr>
<td>Language 1</td>
</tr>
<tr>
<td>V</td>
</tr>
<tr>
<td>[Access Routines]</td>
</tr>
<tr>
<td>V</td>
</tr>
<tr>
<td>------------------------</td>
</tr>
<tr>
<td>Database 1</td>
</tr>
</tbody>
</table>

<p>| Application n .......... |
| V                      |
| AI system control      |
| module                 |
| V                      |
| [Programming]          |
| Language n             |
| V                      |
| [Access Routines]      |</p>
<table>
<thead>
<tr>
<th>V</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database n</td>
</tr>
</tbody>
</table>

Figure 1. The data management environment of AI systems.
languages (LISP, PLANNER, CONNIVER, PROLOG, LOGO, etc.) and knowledge representation languages (KRL, FRAIL, Conceptual Dependency, etc.), but the phrase AI database is much less widely used, and at that, has little in common with the theoretical attitude towards databases (cf., though, Wong & Mylopoulos (1977), where the relationship between data models and predicate calculus-based AI representations is studied). In short, the role of data structures and management in AI applications has played a subservient role to language design. A problem-oriented data model is needed to help the implementors of an AI system define its components and parameters, to help users in their interaction with such a system and, last but not least, to facilitate comparison and evaluation of such systems.

All the above considerations have convinced us of the necessity of a unified data model for AI applications.

The question may be asked: why not use one of the existing data models (either network, hierarchical or relational one)? The answer lies in the peculiar requirements of the AI research on its data model. The latter must contain the appropriate tools for supporting a system which is able to react intelligently to its input. Thus, the ability to conduct "Gedanken-experiments" in building plans must be supported: a basic operation in an AI system is to evaluate the consequences of a certain assumption without changing the "main" database content and subsequently remove the result of the inference-making process from the database. The automatic deduction itself is another requirement: the ability to infer facts from facts (on the basis of user-supplied deduction rules), from simple closures in "isa" hierarchies to finding antecedents for pronouns in natural language text processing, must be strongly supported, too.

Another major requirement for an AI data model is to be able to support the environment of learning, which gradually becomes one of the key issues in AI. The corresponding tools must enable dynamic changes of programs at run time, as well as of data and data organization.

Practically none of the above requirements of AI systems (and more could...
be mentioned) are met by the general database management systems; nor is there a sign that the situation will change in near future.

All the three standard data models are (or can be) obtained as a mapping from a conceptual entity-relationship model, which recognizes only such categories as entity sets and relationships, distinguishing the latter only functionally, as one-to-one against many-to-one against many-to-many relationships. This metalanguage does not seem to possess the expressive power necessary for reflecting the world of AI applications. Even the introduction of “built-in” or system relationships, such as “isa”, does not, by itself, suffice to upgrade this expressive power (though we introduce the isa-hierarchies in our model).

There is a need for integration between AI and database theory, so that a model can be created that will define the role and the way of use of data in AI systems and programming environments. First of all, we must understand how to define a model of defining data structures, creating instances of a database, using the data and studying the influence of the data on the system activity.

Let us discuss the guidelines for the design of the data model for AI applications. The model should possess the following characteristics:

1. It must be as conceptually simple as possible; in other words, the least number of elementary notions is the best;

2. It must be as universal as possible; the model must be independent from the implementation domain, from the psychological model of cognition underlying the design of a specific AI system and of the approach to knowledge representation (the choice of language);

3. It must allow for flexibility: every element of the data organization and manipulation (schema, retrieval programs, spontaneous activity routines, etc.) should be subject to modification at run time.

Among the peculiarities of AI systems as regards their data manipulation are:

1. The ability of the system to undergo spontaneous changes through non-demand-based activity; this feature allows the system, among other things, to save the information it derived at all levels of representation;

2. Coexistence in the database of data obtained through following the deduction chain from several (possibly, contradicting) assumptions which leads to the ability of simultaneously checking several plans with minimal redundancy of data;

3. The ability of the system to deduce facts not stated explicitly in the database, in part, based on the built-in tools for representing hierarchies of static descriptions (the "isa" hierarchies are an example of such).

4. The necessity of working in multiple environments (or modules) corresponding to the various specific subtasks in an AI system (cf. morphological, syntactic and semantic parsing, inference making and text generation as typical and largely intertwined modules of a natural language processing system).

The Proposal

Definitions. The basic and the most important concept in our proposal is that of a line (in our opinion, the word "record" is much overused). Much effort was devoted to enabling most of the concepts in this model to be represented through this notion.

Definition 1. A LINE is an ordered finite set. The elements of this set are called fields. The values of fields are elements of (possibly, named) sets called domains.

The semantics of the domains of fields in a specific line determines a sub-classification of lines. This sub-classification is a value of a special domain called LINE-VARIETY. The elements of this domain will be discussed below.

The two most important line-varieties are as follows:

Definition 2. The TYPE-LINE is a named line whose field values are names of domains. These values are called ATTRIBUTES.

Definition 3. The TOKEN-LINE is a line whose field values are neither names of domains nor names of type-lines.

Each token-line is an instance of a certain type-line, that is there is a many-to-one correspondence between the set of token-lines and that of type-lines. The nature of the correspondence between a token-line L and its corresponding type-line L is as follows:

1) the number of fields in L is equal to that in L.
2) for every field i of l the value of i is an element of the domain of the i-th attribute in L.

Token-lines are unamned but identifiable through the values of their keys, the (groups of) attributes either designated or added by the designer for the specific purpose of line identification. This definition is somewhat circular, but there is no possibility to define a key other than in terms of identifiability of records (cf. Ullman, 1982, pp.12-14).

The notion defined next is one of the two basic tools of representing relations among lines (the other being the static "isa" hierarchy implemented through dedicated data fields in any relevant "object" data lines).

Definition 4. A LINK-LINE is a line whose field values include two or more line-id's of the same kind (i.e. either names of type-lines or key values of token-lines). Instead of names of type-lines the name of a parent node in the "isa" hierarchy can be used (group id instead of individual id).

Since the link-line actually traverses both the realm of type-lines and that of token-lines, we can see that actually we deal with four varieties of lines, determined by the four combinations of answers to the following two distinguishing questions: 1) are the field values of the line names of domains? and 2) are some of the field values of the line line-id's? Table 1 summarizes the results:

Question 2: yes no

Question 1: yes TYLL TYDL
            no TOLL TOLD

Table 1. Line Varieties: TYLL means type-
        link line; TYDL, type-data line; TOLL, token-link line; and TOLD, token-data line.

The data lines are considered to be linked if their id's--either names or key values--appear simultaneously in the fields of a certain link-record. Within this approach the link-lines represent facts of the world for which information is gathered. Automatic inference making (as a result of both user-triggered and spontaneous activity of the DBMS) amounts, therefore, to generating new link-lines in the database at least as much as to modifying the values of token-line fields.

Also, inasmuch as link-lines can be interpreted as introducing a network structure on the collection of lines in the database, this model can be considered an extension of the network model (the definitions given above are, however, free of certain restrictions on the sets and members of, say, the DBTG proposal; also, there is no a priori difference in operations permitted on the two levels, the type and the token one). The two types of networks are, however, distinguished, the first one being referred to as the metanetwork. This metanetwork corresponds to what is usually called schema in the database theory. The above let us treat the database of our model (metaphorically) as a collection of networks or even one (possibly, disjoint) network.

Definition 5. An AREA is a subset of the metanetwork identified by a list of TYDLs and TYYLs that connect them.

An alternative solution would be to allow multiple metanetworks corresponding to areas. These would be components of the highest-level network. Specifying areas is a way of identifying subnetworks in token realm. The notion of area will be useful, for example, in limiting the scope of a certain operation (a search or an update) in the database by limiting the number of type-lines whose corresponding token-lines must be scanned. There can be, however, other scope limitation strategies: cf. the chronological restrictions of limiting the search to k lines last accessed. Since in AI applications it is necessary to store different and often contradicting "possible worlds" in the database, we must provide the model with a tool facilitating the simultaneous existence of different data "versions" and the retrieval according to (or in the CONTEXT of) one such possible world.

If in every such possible world represented in the database we determine the points at which it becomes possible to generate a new context or return to the previous ("father") context, then we obviously speak about a possible world or context) trees. The nodes in this tree are the snapshots of the database status at a certain point in computation (or, theoretically, thought process or plan generation routine), and arcs represent the special relation "be a child context of". As a result of introducing the contexts, we arrange our networks into a higher-level tree structure. Actually, this structure can be a network as well. Our decision to define it as a tree stems, most probably from the influence of the problem-solving techniques in AI, where
An example

Consider a modification of the well-known milieu of the robot Robbie in the blocks world. We would like to suggest the data organization for a research environment in which Robbie moves objects in the world and builds complex objects from them without being prompted by the human experimenter. This model involves two different worlds: the "real" world of objects and Robbie's world, his "understanding" of the former. In real life, that is outside computer modelling, the "real" world does not have any "representation" by itself—only in the eyes of the beholder. In the environment presented the experimentors must simulate the description of this world in order, first of all, to be able to account for "success" and "failure" judgments in Robbie's endeavors.

So, this "real world" is spatially restricted and populated by the movable objects (in this example Robbie's working arm is suspended and is not really a part of the working surface—this will simplify the spatial navigation). The "real" world metanetwork includes a TYDL SPACE (Length, Width)—we assume that its height is practically unlimited, and a TYDL OBJECT (Name, Dimensions, Color, Position), where the domain of Name is {ball, block, box, cube, pyramid} and position is represented through the set of Cartesian coordinates x, y and z. The "real" world (token) network includes, then, the objects in specific positions, the position being the candidate primary key. By definition, the "real" world does not contain relationships, these being a part of a cognitive entity's picture of the universe—by virtue of which there are TYLLs in the "real" world network.

The metanetwork of Robbie's world consists of the descriptions of movable objects: the TYDLs for the Name domain (see above) plus the one for Robbie itself: this would contain RName and RPosition attributes. The important point is that the attribute RPosition, unlike Position, draws its values from the domain (center, center-left, center-right, upper-left, upper-center, upper-right, lower-left, lower-center, lower-right), or, if we suggest "approximate" polar coordinates, with the center of coordinates at RPosition(Robbie), {o'clock, distance}, leading to TYDLs containing such phrases as "Two blocks, 10 o'clock, far", which shows that distance is, again, not measured in absolute terms. An instance of the "isa" hierarchy will be superimposed on this metanetwork, stating, for example, that BALL isa MOVABLE-OBJECT. The TYDL SPACE is an example of an unnovable object.

TYLLs in Robbie's world will include
in(MObject, MObject, Condition), CONTAIN,
ON, UNDER, BEHIND and BEFORE with the same attributes, LEFT, RIGHT, BACK, FRONT, and CENTER with attributes (MObject, Object, Condition). The values of the attribute "condition" are taken from the domain of condition-action rules, specified by the experimenter or derived by Robbie himself, if the environment of the experiment calls for learning (and the data model does not impose any restrictions on this possibility). There is an "isa" hierarchy on these TYLLs: all of them are "isa" links.

Another area in the metanetwork of Robbie's world includes such information as the "antonymity" relationships between, say IN and CONTAIN: TYLL ANTONYM (Link, Link). In addition, such experiential knowledge as whether Robbie is able to move a certain object is represented in TYLL CAN-MOVE (Robbie, MObject), etc. This area contains Robbie's knowledge about the meanings and relationships of objects in his world, thus becoming a kind of meta-metanetwork.

The (token) network of Robbie's world contains information about the presence and whereabouts of specific objects in the world as well as Robbie's descriptions of spatial relationships between these objects as well as his experience in moving the objects around, etc.

The use of the different areas is determined by the nature of the task: thus, when it is necessary to confirm "success" or "failure" of an attempted operation, the "real" world area will be used, while the results of a success will be recorded in Robbie's network area. If Robbie learns from the "real" world procedures that he cannot move a certain block, the information to this effect is recorded in the meta-metanetwork area of Robbie's world, etc. Note that it is not our intention now to suggest the DML for this proposal. We speak only about the data model.

The contexts will be used in Robbie's world when, having to produce a plan for action, Robbie will tentatively change the state of the world (his world, not the "real" one) to find whether this leads to the solution desired. Even in unsuccessful search Robbie may learn certain facts valuable for the future and deserving remembering. This arrangement can be accommodated, for example, in dynamic modification of the static evaluation functions in game playing or remembering plan-building sequences in robotics (even those that did not lead to a viable plan in at the time).
search in a graph has been, as a rule reduced to the search in a tree.

Definition 6. A special node in the network (not the metanetwork!) in which the current "version" of the database is identified is called a CONTEXT-LINE.

A context-line contains the id's of the link-lines representing new facts arrived at since the previous branch-triggering assumption was made. It may also contain information in excess of that needed for database version identification.

The activity of building the tree of contexts sets up a framework for a multiple-purpose representation of the data necessary for calculating the "static" evaluation functions in such applications as games, problem solving, generating plans for robots, etc.

Discussion and Further Problems. The basic proposal for a data model stated above needs elaboration. We have been thinking about adding new specialized domains for every line, such as an indication of conditions that govern the insertion of a new fact into the database or updating existing data. This tool will enable the incorporation of demons or left parts of production system rules. Another line of thought is to introduce more types of restrictions on search, in addition to areas and contexts. One such type may indicate what subset of a line's fields is to be considered in a given environment, thus becoming a tool for implementing the ideas about allocation of attentional resources.

The most important characteristics of the proposal in its present form are first and foremost, the perception of the schema as a part of the data and operation within contexts.

As stated above, the schema is but a marked variety of network, the metanetwork. By defining the schema in this way one gets the advantage of being able to use all the operations defined in the database on the schema as well. The database can retrieve facts about the schema, that is, it "knows about its own existence". If the cost can be afforded in a DBMS, we could introduce the ability of the database to change its own schema through the same DML command as the changes to the data "proper" (most databases that support activity on schemas have special commands for this purpose and do so only within the bounds of fact-finding about the schema, cf., though, the ability of the System R to dynamically add a new column to an existing relation (e.g. Astrahan et al., 1979)). This command will be able to add new type-lines, change the field values and/or names of existing ones, add link-lines, etc. The ability to do so becomes very valuable when we remember that in AI a system may well be required to learn the relevant attributes of objects in its world during its operation. For example, it is a far greater achievement for an AI system to learn that physical objects contain in their descriptions the attribute "material" with the domain {concrete, plastic, obsidian, ...} than to learn that a particular table's legs are made of oakwood. This feature also allows improvements in the database as a result of runtime statistics, and without any specialized commands for schema manipulation.

Currently the systems that implement contexts (or a similar mechanism of automatic backtracking, cf., e.g. the discussion in [Sussman, Winograd & Charniak, 1978]) uniformly proceed from the assumption that the evaluation of concurrent hypotheses is independent. This strategy results in destroying all the changes and extensions carried out in a certain context before returning to the "father" context. This independence assumption seems to lack psychological validity and to be employed mainly due to the implementation preferences. The designers of CONNIER noticed this fault in PLANNER (cf. references above) and made an attempt to overcome it. Still, in our opinion, more powerful and clear tools are needed to make use of the interaction of the processes during concurrent hypothesis evaluation. The idea of parallel processing can be employed here (cf. the application of multiprocessor environment to parsing natural language in Losinskii & Nirenburg, 1982).

In other words, we suggest that it be possible for data updates carried out within specific contexts to be global, i.e. to be "remembered" at a higher level in the context tree. This, in turn, leads to the problem of "update propagation" in the context tree: how are the changes done in one context "passed" to other (both "parent" and "sibling") contexts, and, no less important, what criteria can be chosen for terminating the propagation (specifying a context to which a particular update is context-dependent and not "global" or "inherited"). It may well become a responsibility of the user to specify when and how widely a certain action is context-dependent. The above ability will also lead to the possibility of non-destructive return from a context, with the option of going back there and finding all the previous results intact in that corner of the database.
Conclusion

This paper only gives a short outline of the data model with the help of which AI applications can be expressed. To upgrade the proposal to the level of a DBMS one needs to specify the data manipulation language and the navigation procedures within the database, as well as the mappings of logical records (lines) into physical storage. This is the direction of our further studies.

Bibliography


