A Generic Framework for Expert Data Analysis Systems

Luanne Burns
and
Alexander Pasik

CUCS-163-85

August 23, 1985

Department of Computer Science
Columbia University
New York City, New York 10027

A Generic Framework for Expert Data Analysis Systems

Abstract

DTEX, a generic expert system building tool, attempts to minimize (or in the best cases eliminate) the role of the knowledge engineer by providing the expert with an interactive system that facilitates the knowledge acquisition process as well as automates the generation of an executable expert system. DTEX addresses database analysis domains by providing a framework for communication between database management systems and expert systems. Details of the database manipulation language are not incorporated directly into the knowledge base. The inference and support knowledge of the expert system are isolated from the syntax of the particular DBMS, thus providing a uniform method for the design and execution of expert data analysis systems.

A Generic Framework for Expert Data Analysis Systems

1. Introduction

The emergence of database technology revolutionized data processing by providing a controlled storehouse for high volumes of data, transformable into pertinent information, with respect to varying demands. The database concept offered data independence and therefore, increased data integrity, consistency, and non-redundancy, and also responded readily to data access and manipulation through conventional, algorithmic programming methodologies.

Researchers interested in capturing the essence of expertise developed applications which codified domain-specific knowledge representation and reasoning. Early expert systems work concentrated its efforts on simulating the expert's loosely structured processes of drawing conclusions about a specific domain, using facts concerning that domain along with heuristic rules of thumb (Shortliffe, 1976, Buchanan *et al.*, 1969, Duda *et al.*, 1978, McDermott, 1982). The use of a knowledge engineer as the bridge between the expert and the technology was essential to these knowledge-based systems.

As the technologies matured, it became apparent that the database and expert system formalisms each possessed qualities that held valuable potential for the other. The database world, although able to store vast amounts of information, needed trained analysts to interpret the data in meaningful ways. Conversely, expert systems were limited to domains in which a given instantiation could be represented in a small working memory. Because of this shortcoming, and the limitations of available tools and technology, pragmatic, cost-effective use of expert systems in real-world problem domains was restricted to a small class of problems (Kellogg, 1982).

The outgrowth of this has been the development of systems which synergistically merge conventional database technology with expert systems, such as ACE (Stolfo and Vesonder, 1983). The quantity of information available to the expert system is greatly increased by the existence of a database management system (DBMS). In addition, automatic analysis of an existing database is made possible by the inferential capabilities of the expert system.

The proliferation of expert systems is inhibited by the difficulty in acquiring knowledge and reasoning from appropriate human experts. This requires the use of a knowledge engineer, who, by working closely with an expert, attempts to organize and codify the expertise. The need for this knowledge engineer is a bottleneck in the knowledge acquisition process. Domain-independent expert systems such as EMYCIN (van Melle, 1979) and Hearsay-III (Balzer *et al.*, 1980) attempt to separate the methodology from its specific applications, substantially simplifying the work of the knowledge engineer. Nevertheless, building an expert system with these tools still requires programming proficiency. DTEX, a generic expert system building tool, attempts to eliminate the role of the knowledge engineer by providing the expert with an interactive system that facilitates the knowledge acquisition process as well as automates the generation of an executable system.

DTEX addresses database analysis domains by providing a framework for communication between DBMS's and expert systems. Data analysis represents a large class of problems with solutions that can be automated using this formalism. These expert systems would be applicable immediately.

Unlike ACE, DTEX does not incorporate details of the database manipulation language directly into the knowledge base. The inference and support knowledge of the expert system are isolated from the syntax of the particular DBMS, thus providing a uniform method for the design and execution of expert data analysis systems. This technique can greatly accelerate the development of data analysis systems.

Howard and Rehak describe KADBASE (1985), a system which provides a general query language for the communication between specific expert systems and database systems. One driver from the expert system to KADBASE and one from the DBMS to KADBASE are necessary. DTEX differs by providing a formalism for the construction of the expert system; additionally, because DTEX supplies its own query language for communication between any DTEX-built expert system and a database, a single driver suffices for a given DBMS.

2. Overview of DTEX

The DTEX system consists of two modules: DTEX/KA for knowledge acquisition and DTEX/PSE for executing the problem solving strategies and explaining the behavior of the generated expert systems (Pasik *et al.*, 1985). DTEX/KA acquires objects and relationships between these objects directly from a human expert in the given domain. The objects fall into three categories: *properties* which correspond to the data a given system will base its inferences upon, *characteristics* which describe the overall goals of the system, and *conclusions*. DTEX acquires *relationships* between the properties and characteristics and between characteristics and conclusions. These may represent the causal knowledge in a domain.

Once DTEX has acquired this abstract representation of the problem domain, it compiles the knowledge into a decision tree, inferring the control knowledge from an analysis of the relationships. DTEX/PSE is an interpreter of these expert systems generated by DTEX/KA (see Figure 1).

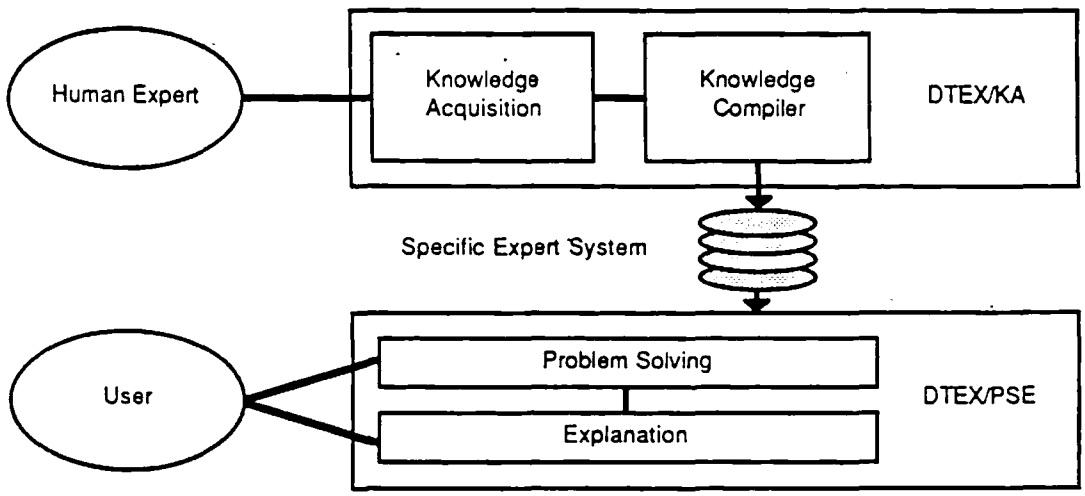

Figure 1: Structure of DTEX

In the acquisition of a property, DTEX must be given a description of a method for its calculation. It acquires a formula for each property which, in the degenerate case, can be "ASK", implying that the value will be supplied by the user. A variety of operators are built-in including those which require database access.

The in-depth explanations provided by DTEX are concerned with the inference steps in the decision tree. The support knowledge used to generate these explanations is inferred from the relationships acquired. Because database access is restricted to occur in the evaluation of a property, the explanation module need not be concerned with attempting to explain mundane data access routines.

An example of an expert data analysis system in DTEX is a system designed to make recommendations concerning the allocation of disks to cache controllers on a large computer system (Guttman, 1984). In this problem, a disk volume is considered based on several of its properties ranging from its data access patterns to the type of information stored on it. From this information, intelligent decisions must be made regarding whether the disk should be placed on a cache controller. The properties would include the *volume type*, the *percent busy*, the *read/write ratio*, and the *read hit percentage*. Problem characteristics that must be addressed include the *likelihood of reading from the disk*, *writing to the disk*, and the *likelihood that successive reads are to sequential records*. A relationship that DTEX would acquire between a property and a characteristic would be that *percent busy less than 60% implies that reading the disk is very unlikely*. A portion of the inference structure created by DTEX, along with the support knowledge inferred from the above relationship, is shown in Figure 2.

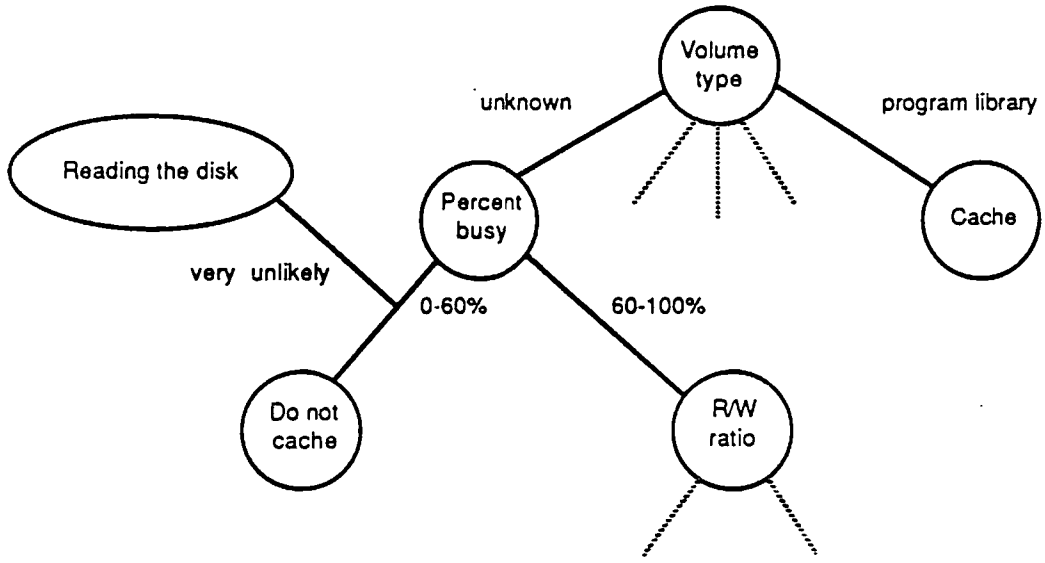

Figure 2: Part of a decision tree

3. Interfacing databases with DTEX

There are numerous applications that query and report against existing databases. Experts peruse the output, interpreting its contents and making decisions. This might include analyzing a fairly static database, such as one containing census information, or drawing conclusions about a transaction database, as in ACE's analysis of trouble reports. Using DTEX, the expert can tailor the knowledge base component to automate these tasks. To the DBMS, the expert system appears as a user. It is, however, an automated database user that knows not only what to retrieve from the database, but also how to interpret the results of its inquiries.

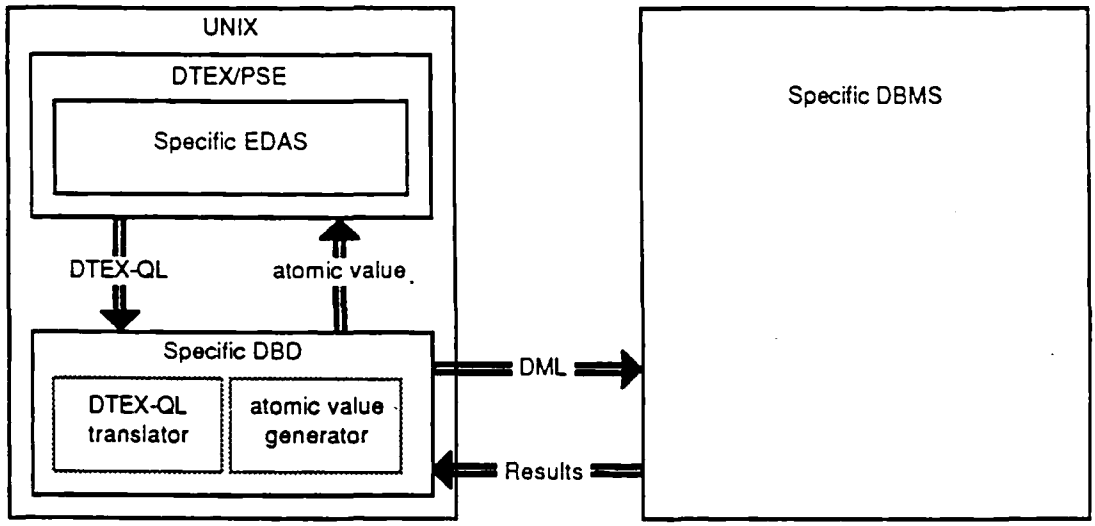

Figure 3: DTEX/DB Interface

DTEX can be used to create expert database analysis systems (EDAS) such as the intelligent information retrieval systems described by Zarri (1984). Figure 3 represents the architecture of a DTEX EDAS. It operates in the UNIX™ environment as two concurrent processes, DTEX and a database driver (DBD), communicating via a pipe. DTEX's problem solving engine generates statements in a query language (DTEX-QL) which are meaningful to the driver process; the driver, in turn, translates these statements into the proper syntax and semantics of the data manipulation language (DML) for the DBMS to which it interfaces. It is important to note that the implication of this design is that one driver for each DBMS with which a DTEX EDAS would interface is required but constancy is maintained with respect to the manner in which a DTEX EDAS itself interacts with any database; it always formats it queries using the standard DTEX-QL and relies on the driver to perform the mapping to the appropriate DML.

The DBMS then returns query responses to the driver, just as it would to any other user; the driver interprets the results and returns an atomic value to the problem-solving engine. The result is atomic because, recalling the structure of DTEX problem-solving, the value of a property is determined by either asking the user or through evaluation of some formula. It is in this formula that an operation against the database might be included (e.g. MAX, MIN, AVG,... for an attribute of a particular relation). The atomic value is inserted into the formula and evaluation continues. DTEX then uses the result of the evaluation to direct its path in the decision tree.

Although DTEX is restricted to using scalar values in its inferences, tabular results from database queries can be displayed upon reaching a conclusion in the decision tree. In this way, DTEX can generate reports reflecting its intelligent analyses.

To make the interface complete, it is necessary for DTEX, in the knowledge acquisition module, to learn about the external database tables it will be manipulating and, if any exist, the tables that will be derived from them through use of relational algebra's traditional set operations. In the knowledge base, therefore, should be stored the name of the table and the type (external or derived). If it is a derived table, the method for deriving it should be acquired. This would include the operation (select, project, or join), the table(s) from which it is derived, and the operation criteria (the joining or selection condition, the projection attributes). Also the intension of all tables (Date, 1981) (i.e. the name and datatype of the attributes, key fields and other permanent, time-independent information) should be resident in the knowledge base of the expert system. Specifying the external tables could be done in several different ways. One, requiring no concurrently running database driver during knowledge acquisition, is to have the expert explicitly state the table name and attribute information for each external table. Another is to have a separate driver process interfacing the knowledge acquisition module to the DBMS; this would allow the expert to simply specify the table name which would, in turn, direct the driver to retrieve the intensional data and store it in the knowledge base.

During the acquisition of a derived table, the expert may specify virtual columns. These columns correspond to attributes which do not have their values explicitly stored but rather, when referenced, are calculated by some formula. This logical field of a tuple is materialized from a computation performed on one or more of the stored field occurrences (Date, 1981). Thus, iterative operations on data in a table may be implicit in the table's definition. For

example, an employee table may contain attributes such as gross pay, total deductions, and net pay. Net pay is a virtual field; its value is determined by gross pay minus total deductions. Because the inference structure of a decision tree prohibits iteration, virtual columns are essential for performing operations on all tuples in a table.

DTEX's knowledge acquisition facility acts as the knowledge engineer during the construction of an EDAS. Nevertheless, the domain expert still needs the assistance of a database professional in order to formulate the appropriate relational queries in DTEX-QL. An alternative, albeit lower level, approach towards the interface is available via DTEX's *prog* operator. This operator serves as a back door in DTEX; a formula including a prog executes an arbitrary user program to calculate the value of a property. Thus, the domain expert may not need the assistance of the database professional if a script for the appropriate data access can be executed through the prog operator.

The DTEX/DB interface can be illustrated by the following example. Suppose the relational database concerned has the following relations:

```
table T1 -- attributes a, b, c, d
table T2 -- attributes d, f, g, h
```

Tables that will be derived are as follows:

```
table D1 -- operation:    JOIN
                          from tables: T1, T2
                          condition:  T1.d = T2.d
                          attributes:  a, b, c, d, f, g, h

table D2 -- operation:    SELECT
                          from table:  D1
                          condition:  b > p1, where p1 represents a property
                          attributes:  a, b, c, d, f, g, h

table D3 -- operation:    PROJECT
                          from table:  D2
                          condition:  a, c, g (projected attributes)
```

For simplicity, assume the datatype of all attributes is the same. The above schematic information is acquired during the creation of the EDAS using DTEX's knowledge acquisition module. DTEX uses lazy evaluation, in that the derivation of a derived table does not take place until it is actually referenced, either directly or indirectly, in a formula. Let us suppose that DTEX, executing a given EDAS, has values for properties p1, p2, and p3 and must now determine the value for p4, defined by:

```
p4 = AVG(D2,g)
```

which is equivalent to the average of all g attributes in table D2. DTEX recognizes that AVG is a table operation and control is passed to the query formation stage of the problem solving engine. It recognizes that indirect reference is made to tables T1, T2,and D1; all are necessary to form D2, which is referenced directly. It assembles the appropriate DTEX-QL commands which are then piped to the driver process as follows (assume the value of p1 is 5037):

```
D1 = JOIN(T1,T2, T1.d = T2.d)
D2 = SELECT(D1, b > 5037)
return AVG(D2, g)
```

The driver receives the DTEX-QL input and transforms it into the DML of the DBMS, SQL (International Business Machines Inc., 1984) and MISTRESS (Rhodnius Inc., 1982) in the following examples:

SQL DBD output

```
select   avg(T2.g)
         from T1,T2
         where T1.d = T2.d
         and b > 5037
```

MISTRESS DBD output

```
select   from T1, T2
         join where T1.d = T2.d
         insert into D1

select   from D1
         where D1.b > 5037
         insert into D2

select   avg g from D3
```

The driver, acting as a database user, sends these commands to the DBMS and monitors the resulting data streams. Upon receiving the final response, it releases the derived tables from the database and returns the atomic value to the problem-solving engine. Derived tables are released because, as DTEX proceeds, the value for properties used in the conditional phrase of the derived table specification may change before the next invocation and reference to that table. Thus, derived tables are generated each time in the lazy evaluation method when they are directly or indirectly referenced, and are subsequently released before control is returned to DTEX. DTEX continues, using the value now assigned to p4; hence, it not only poses a query to the database, but it proceeds to interpret the result in its subsequent inference processes.

DTEX's database operations can be divided into two classes: relational algebra operations which act on tables to generate new tables, and reduction operators which act on tables and yield atomic results. Only these atomic values are used in the inference process; reduction operators are therefore essential in providing adequate access to the database. These include operations ranging from data extraction from a particular tuple in a table (e.g. LOOKUP(T,a1,v1,a2) which returns the value in table T of attribute a2 such that attribute a1 equals v1) to global operations on all tuples in a table (e.g. COUNT(T) which returns the number of tuples in T). The following is a list of reduction operators in DTEX.

COUNT(T)	number of tuples in T
AVG(T,a)	average of all values of attribute a in T
SD(T,a)	standard deviation of all values of attribute a in T
SUM(T,a)	sum of all values of attribute a in T
MAXT(T,a)	maximum value of attribute a in T
MINT(T,a)	minimum value of attribute a in T
LOOKUP(T,a1,v1,a2)	value of atrribute a2 in T such that a1=v1

DTEX also provides non-table operators. Relational algebra, reduction, and non-table operators can be combined to determine values for properties in DTEX/PSE. The following example illustrates the distinction between types of operators. Consider a property *p* defined by a formula which depends on another property *q*:

```
p = IF(((AVG(T,att) - (SD(T,att) * 2)) > q), FAIL, PASS)
```

- AVG and SD are table reduction operators.

- the references to table T imply relational algebra operators in that if T is a derived table, derivation of all its ancestors must occur before the reduction operators can be applied.

- IF, -, *, and > are non-table operators.

The expression translates to English as follows: If the difference between the average and twice the standard deviation of table T's attribute att is greater than the value of property *q*, return the symbolic constant FAIL, otherwise return PASS.

DTEX's table operators provide the ability to extract information about a particular entity occurrence as well as summarize data about an entity type. This fuctionality, along with the implicit iteration provided by virtual columns, allows DTEX to address a host of database analysis tasks.

Referring to the disk cache controller example of the previous section, several of the properties would be defined in terms of formulas involving database access. A database may exist which provides some of the needed information easily while other data must be calculated explicitly. For example, one table may have statistics on each volume including its percent busy. On the other hand, the read/write ratio and read hit percentage of a volume may be calculated from a large table of all disk accesses on a volume created by a trace package. Both of these kinds of values can be extracted and returned to DTEX's inference engine for the problem solving task to continue.

4. Current implementation

An internal version of this system has been implemented and tested; all components are written in OPS5 (Forgy, 1981) and include sets of productions acting as a primitive database manager, the database driver, and the query generator. Tables, their tuples and individual elements, are stored as working memory elements that the DBMS rules alone manipulate. Internal tables have their elements explicitly stored while derived tables consist of tuples, or portions of tuples, extracted from other derived or internal tables. The query formation stage of problem-solving is given control when a table operation is detected during formula evaluation. From the operation specification, it determines which tables must be derived. It nests requests

for all ancestor tables of the one explicitly stated, until a necessary parent relation is determined to be internal. These requests are the OPS5 equivalent of DTEX-QL. The role of the driver in this prototype is somewhat passive in that it need not map the requests into a specific data manipulation language for the OPS5 data manager. Its job, in this stage, is merely to transfer control to the DBMS. The DBMS creates the tables, one by one, beginning with the one(s) whose derivation is directly from a user-specified internal table, and works its way to the table specified in the original table operation. After all necessary tables have been derived, the DBMS rules pass control back to the driver which then formulates the query representing the actual table operation. This is passed to the DBMS which performs the operation and returns an atomic response. The driver, now playing a more active role, takes control by re-incorporating the returned response into the formula evaluation. Problem solving continues, inferences are made, and DTEX continues to traverse the decision tree.

5. Conclusion

Consultation expert systems have not been accepted in many domains because the question of responsibility is an issue. Database analysis represents a large class of problems where these issues are not as crucial. Here, new applications are needed to help businesses intelligently analyze the ever-increasing amount of information that needs to be interpreted. Expert data analysis systems would thus be beneficial immediately. DTEX provides a pragmatic approach to the development of such systems.

References

Balzer R., L. Erman, P. London, and C. Williams. *Hearsay-III: A Domain-independent Framework for Expert Systems*, pages 108-110. AAAI, Stanford University, August, 1980.

Buchanan, B.G., Sutherland, G.L., and Feigenbaum, E.A. Heuristic DENDRAL: a Program for Generating Explanatory Hypotheses in Organic Chemistry. *Machine Intelligence*, 1969, *4*. Meltzer and Michie, editors, Elsevier, New York.

Date C. J. *An Introduction to Database Systems*. Reading, Massachusetts: Addison Wesley 1981.

Duda R. O., J. Gaschnig, P. E. Hart, K. Konolige, R. Reboh, P. Barrett, and J. Slocum. *Development of the PROSPECTOR Consultation System for Mineral Exploration*. Technical Report SRI Projects 5821 and 6415, SRI International, Inc., 1978. Final Report.

Forgy C. L. *OPS5 User's Manual*. Technical Report CMU-CS-81-135, Department of Computer Science, Carnegie-Mellon University, July 1981.

Guttman P. T. *Methods for the Deployment of IBM 3880 Model 13 Cached Storage Controllers*, pages 594-604. Proceedings of the Computer Measurment Group XV, December, 1984.

Howard H. C. and D. R. Rehak. Knowledge-Based Database Management for Expert Systems. *SIGART Newsletter*, April 1985, *92*, 95-97.

International Business Machines Incorporated. SQL/Data System Terminal User's Reference for VSE.

Kellogg C. *Knowledge Management: A Practical Amalgam of Knowledge and Data Base Technology*, pages 306-309. AAAI, Carnegie-Mellon University, August, 1982.

McDermott J. R1: A Rule Based Configurer of Computer Systems. *Artificial Intelligence*, September 1982, *19(1)*, 39-88.

Pasik A., J. Christensen, D. Gordin, A. Stancato-Pasik, and S. Stolfo. *Explanation and Acquisition in Expert Systems Using Support Knowledge*. Technical Report, Columbia University, 1985.

Rhodnius Incorporated. Mistress: The Query Language.

Shortliffe E. H. *Computer-based Medical Consultations: MYCIN*. New York: American Elsevier 1976.

Stolfo S. J., and G. Vesonder. *ACE: An Expert System for Telephone Cable Maintenance*. Technical Memorandum, Bell Labs, April 1983. Short version.

van Melle W. *A Domain-independent Production-rule System for Consultation Programs*, pages 923-925. IJCAI, Tokyo, Japan, August, 1979.

Zarri G. P. *Intelligent Information Retrieval: An Interesting Application Area for the New Generation Computer Systems*. Institute for New Generation Computing, Tokyo, Japan, November, 1984.