Verification and Validation Issues in Expert and Database Systems: The Expert Systems Perspective

Frans Coenen
Department of Computer Science, The University of Liverpool, Liverpool L69 3BX, England.
Tel: 0151 794 3698, Fax: 0151 794 3715, email: frans@csc.liv.ac.uk

Abstract

This paper is directed at two central objectives. The first is to identify and establish areas of overlap between the expert and database system domains. The second is to present a view of existing and ongoing work within the expert systems community concerning the Verification and Validation (V&V) of rule bases. This review is directed towards the database community with the express aim of identifying possibilities where expert system V&V knowhow may also be of value to the database community, especially with respect to the identified areas of overlap.

1. Introduction

This paper seeks firstly to establish some common ground between the Validation and Verification (V&V) of databases and rules bases, and secondly to identify possible areas where rule base V&V techniques may be applicable with respect to database systems.

There are, in the author's view, three main areas where the domains of databases and expert systems interconnect. The first is concerned with the generally acknowledged observation that the relational database model can be expressed in terms of clausal form logic. The second is in the field of deductive databases which seek to extend the functionality of the relational model by allowing predicate calculus rule constructs to be applied to data. The third is in the area of active databases where expert system style sets of rules are used to automatically update records given some "triggering" event. A possible fourth candidate may be the link between Constraint logic Programming and constraint databases. Further details concerning these areas of overlap are presented in Sections 3, 4, 5 and 6, after a brief defining overview of expert systems given in Section 2. Where appropriate rule fragments are presented using the PROLOG expert systems programming language.

The remainder of the paper is set out in a review style with each Section covering a sphere of interest within the expert systems V&V community, namely: errors and anomalies in rule bases, error detection and remedial action. Each Section commences with a brief outline of the "state of the art", including significant references, and ends with a short discussion of areas which may be of relevance to the database community.

2. Expert Systems

Expert systems typically comprise a rule base and an inference mechanism. Rules are generally described using predicate calculus, and more precisely a simplified version referred to as *clausal form*. Clausal form logic constructs are typically expressed as follows:

$$if < antecedent > then < consequent >$$

where the antecedent part comprises one or more propositions/predicates which, if they evaluate to `TRUE`, will establish the proposition/predicates contained in the consequent. Alternatively a production rule format, popular in many expert systems environments, may be adopted in which case the antecedent part tends to be referred to as the *condition* part and the consequent the *action* part:

$$if < condition > then < action >$$

PROLOG uses a special clausal form known as *horn clauses*:

$$\begin{array}{ccc} < consequent > & :- & < antecedent > . \\ (< action > & :- & < condition > .) \end{array}$$

If a clause has no consequent part the antecedent part is considered to be "unconditionally" true. In such cases the clause is usually referred to as a *fact*.

The principal advantage of clausal forms is that reasoning can be implemented using a technique referred to as *resolution* ([26]). Using this technique an *inference engine* attempts to satisfy the condition/antecedent part of a rule by

films		
name	director	actor
the_trouble_with_harry	hitchcock	gwenn
the_trouble_with_harry	hitchcock	forsythe
the_trouble_with_harry	hitchcock	macLaine
cries_and_wispers	bergman	anderson

...

location		
cinema	address	telephone
odean	lime_street	123456
philharmonic	hope_street	654321

...

whatsOn		
cinema	film	start_time
odean	cries_and_wispers	20.30
philharmonic	the_trouble_with_harry	20.15

...

Table 1. Example relational database tables

replacing it with facts and/or other rules. This process continues until only unconditionally true facts remain, it which case the condition part is said to "succeed". If suitable facts are not found the condition is said to "fail". The process of finding a matching substitution which will make a condition part of a rule equal to an action/consequent part of a rule (or a fact) so that the resolution can be applied is called *unification*. Thus the engine moves from "if patterns" to "then patterns" using the if pattern to identify appropriate substitutions for the deduction of a new antecedents. This resolution mechanism is referred to as *forward chaining* (i.e. *goal driven* searching). The reverse is *backward chaining* (i.e. *data driven* searching) where a rule based system can form a hypothesis and use the rules to work backward towards hypothesis-supporting assertions.

As a result of the principle of resolution, rules in a rule base can be arranged in a hierarchy comprising a top level (root) rule, any number of intermediate rules and a set of leaf rules (facts). In practice there may be several root rules, each node in the hierarchy may have any number of branches and the hierarchy may be extremely unbalanced. The satisfaction of the root rule can then be likened to the satisfaction of a compound query couched in terms of a number of sub-queries such that the result of one sub-query acts as input to another sub-query.

As with main stream software engineering V&V of rule bases has long been a concern in the expert systems community. Early work, originating at the commencement of the 1980s and now well established, sought to establish the nature of errors and anomalies that required detection. As a result many approaches and techniques were proposed to aid in the identification/prevention of such errors and anomalies ranging from design methodologies to V&V tool sets.

```
films(the_trouble_with_harry,hitchcock,gwenn).
films(the_trouble_with_harry,hitchcock,forsythe).
films(the_trouble_with_harry,hitchcock,macLaine).
films(cries_and_wispers,bergman,anderson).
...

location(odean,lime_street,123456).
location(philharmonic,hope_street,123456).
...

whatsOn(odean,cries_and_wispers,20.30).
whatsOn(philharmonic,the_trouble_with_harry,20.15).
...
```

Table 2. PROLOG fact base

Some of the proposed techniques and approaches have been adopted by industry, while others serve as markers in the chronology of rule base V&V research. Work is still continuing in this area. Further current work is also concerned with the refinement of rule bases, subsequent to the detection of errors, with the emphasis on tools to support automated or semi-automated refinement. It is also appropriate to note here that there have been a number of significant ESPRI projects in the field - VIVA, VALID, VITAL ([3]).

3. Rule Bases as Relational Databases

In the relational model (Codd [7]), a database is a specification of a set of relations which can be interacted with using a query language. As such a relational database can be treated as a special case of clausal form logic in which the tuple is regarded as a simple clause with constant arguments. Consider the example relational database given in Table 1 where we have three tables (relations) `films`, `locations` and `whatsOn`. This could be implemented as a series of PROLOG predicates or facts (Table 2). In Table 2 groups of facts are gathered together to form a "table" (in the relational database sense) using a unifying predicate (relation) name. Each line in the PROLOG *fact base* is then synonymous with a record or tuple in a relational database table.

To interact with a relational database we express queries such as the following (SQL) query:

```
SELECT a.films b.director
FROM whatsOn.a films.b
WHERE a.cinema = odean and a.film = b.film
```

The same query can be expressed in logic, using PROLOG syntax, as follows:

```
select(Film, Director):-
    whatsOn(odean,Film,_),
    films(Film,Director,_).
```

Note that as with relational Database Management System (DBMS), expert systems environments and programming languages, such as PROLOG, typically also allow for the addition and deletion of facts. However, not all aspects of relational databases map neatly into clausal form logic. One particular problem is that the relational model is founded on the *Closed World Assumption* (CWA) - if a relationship is not known to be true then it is assumed to be false. This is not supported by pure logic, however CWA does correspond closely to the way in which languages such as PROLOG handle negation (i.e. *negation by failure*).

4. Deductive Database Systems

Deductive database systems provide mechanisms for managing knowledge as well as data. They are a natural extension to the relational model. As such deductive databases are able to derive new facts using existing information explicitly stored in the database [24]. This is achieved by generalising the type of information that may be stored in the databases. In addition to simple facts it is also possible to store expert system style rule sets. This then produces a database system with similar properties to that of a logic programming environment. For example, with respect to the PROLOG "relational database" presented in Table 2, we could store a rule of the form:

```
select(Cinema,Director,Film):-
    whatsOn(Cinema,Film,_),
    films(Film,Director,_).
```

which can be used in a number of different ways according to the manner in which the arguments are *instantiated*:

1. Return all films and their associated director currently being shown at a all cinemas contained in the database (`select(X, Y, Z)`).
2. Return all the films, and the associated directors, currently being shown at a particular cinema (e.g. `select(odean, Y, Z)`).
3. Return all the films, and cinemas where they are showing, given a particular director (e.g. `select(X, bergman, Z)`).
4. Return all the cinemas where a particular film is showing and the director of the given film (e.g. `select(X, Y, cries_and_wispers)`).
5. Return all films by a particular director showing at a particular cinema (e.g. `select(odean, bergman, Z)`).
6. Return the name of the director of a particular film showing at a particular cinema (e.g. `select(odean, Y, cries_and_wispers)`).
7. Return all the cinemas where a particular film with a particular director is showing (e.g. `select(X, bergman, cries_and_wispers)`).
8. Confirm that a particular cinema is showing a given film with a given director (e.g. `select(odean, bergman, cries_and_wispers)`).

The distinction between expert systems and *expert database systems* is that the primary concern of the latter is the management of large amounts of data with the manipulation of that data as a secondary concern [20]. Expert systems, on the other hand, are more concerned with the manipulation of "smaller" amounts of data, and more complex amounts of knowledge, with the emphasis on manipulating that data in a manner that reflects the mode of working of a domain expert. The distinction is a fine one. The ratio of facts to rules may be a useful indicator to distinguishing between an expert system and an expert database - an expert systems will (generally) contain many more rules than facts while an expert database system will contain many more facts than rules.

5. Active Databases

Active databases support the the automatic triggering of updates in response to internal or external events. This is generally achieved in an "expert system" like manner where forward chaining of rules is used to accomplish the update. Rules typically follow the ECA (Event-Condition-Action) format of:

```
on <event> if <condition> then <action>
```

The similarity between this and expert system style rules, especially in the case of production rule systems, is self evident. The distinction is that in a rule base the triggering event is user supplied rather than expressly included. In a sophisticated active rule base the action part of the rule may entail calls to further rules in a manner identical to that supported by experts system style rule bases. The overlap here is so close that the rule set that forms a component of an active database is generally referred to as a *rule base* [2].

Active database behaviour has been applied to the relational model [27], the deductive model [31], and the Object Oriented model [19] of database systems.

6. Constraint Databases

Constraint databases [18] are an extension of research into CLP [17]). CLP combines the advantages of logic programming (declarative, semantics, nondeterminism and relational form) with efficient constraint solving by introducing richer data structures. As such it can be viewed as a generalisation of logic programming where unification (as used to support PROLOG) is replaced by constraint handling in a constraint system [10]. A particular advantage is considered to be that it supports "consistency techniques" [16], based on the concept of *a-priori* pruning, to perform intelligent searches of decision trees. The technique supports constraint propagation in such a way that it reduces a-priori the search space, thus limiting the computational time required.

The basic idea of constraint databases is to replace the notion of a tuple in a relational databases by that of a generalised tuple, i.e. a conjunction of constraints. For example given a tuple:

$$(a, \ ..., \ an)$$

this can be regarded as a generalised tuple of the form:

$$(xa = a1) \wedge ... \wedge (xn = an)$$

Of course this model also needs to be supported by an appropriate "constraint database" query language.

This incorporation of CLP in a database model, where CLP can be viewed essentially as an extension to logic programming may be viewed as another area of overlap between the expert systems and database domains. The use of constraints is also recognised as an important tool for database integrity checking. In early work in this area constraints were expressed in terms of clausal logic. More recently much more sophisticated constructs have been adopted.

7. Errors and Anomalies in Rule Bases

Broadly a rule base can be "wrong" in two ways, it may be structurally flawed or logically flawed. This is a distinction akin to the difference in traditional software engineering between the terms verification and validation which Bohem ([4]) neatly defines as follows:

- Verification: "Are we building the product right?"
- Validation: "Are we building the right product?"

We can also make a distinction between structural errors and structural anomalies. Errors are clearly undesirable in that they will adversely effect the operation of a rule base; anomalies in contrast may not necessarily represent problems in themselves, but rather symptoms of genuine errors.

Much work has been done to establish and classify the nature of the structural errors and anomalies that may be present in rule bases (for example Ayel and Laurent [3] and Preece and Shinghal [25]). In the author's view the most straight forward manner to classify structural errors/anomalies is to make a distinction between errors concerned with *inference* and those concerned with the *design* of the rule base.

Inference errors/anomalies are concerned with the connectivity of the rule base. A rule in a rule base is connected if it has at least one downward connection and one upward connection. If a rule has neither it is unconnected, if a rule has only upward connections it is a leaf rule and if it has only downward connections it is a root rule. A rule that has no upward or downward connections is sometimes referred to as a *redundant rule*, while a rule that is not a leaf rule but has no downward connection is sometimes referred to as a *dead end rule* (in forward chaining systems). Under the heading of connectivity we can also include auxiliary rules; these are defined as rules that have only one upward and one downward connection and therefore may appropriately be subsumed into some other rule. Circularity, an urgent problem in some systems, is also considered to be an inference error.

Design errors/anomalies include subsumption, duplication and inconsistency. Subsumption is usually defined as the situation where one rule is a more specialised case of another. For example the antecedents of two rules are identical except that one antecedent includes one or more additional propositions, and the consequents are also identical, then subsumption exists. Duplication, where the antecedents and consequents of two rules are identical (except perhaps in the ordering of predicates) is then a specialised form of subsumption. Inconsistency then describes the situation where two (or more) rules result in contradictory consequents, i.e. where two rule antecedents are identical but their consequents are not co-tenable. Where the antecedents of two rules are such that subsumption may exist but the consequents are different subsumption and inconsistency are considered to exist together.

A similar classification of logic and design errors and anomalies can be attributed to the rule sets found in deductive and active databases, and relational databases expressed in clausal form. For example circularity has been identified as a problem in active database sets [1, 19]. Although active databases display all the problems of experts system V&V, the issues are sharpened by the consequences of a flawed distributed active database.

For a more detailed discussion concerning the errors and anomalies that may occur in rule bases interested readers are referred to [8]. It should also be noted that although V&V is an issue in database systems, the greatest concern is to do with integrity of the data: the permitted types and ranges of items, possible conflicts and contradictions between fields and the inter-dependencies between tables and fields.

8. Error Detection

There are two main approaches to identifying logical errors in rule bases. The first and most obvious approach is to exercise a set of appropriate test cases in manner synonymous with traditional software testing. The operation of the rule base can then be said to be "correct" if the results produced compare favourably with those suggested by a domain expert. This approach has been adopted by many practitioners [11, 23]. Work has also been done on the automatic generation of test cases [6]. In the second approach information concerning integrity constraints (typically expressing incompatibilities between inputs and/or outputs) that are known to exist with respect to a particular domain are incorporated into the rule base and test cases generated which attempt to break these constraints. In this case a rule base is said to be correct if for all non-contradictory inputs no contradictory outputs result. Consistency checking techniques have been extensively studied [22, 14]. Of course, as with traditional software testing, using either technique, the rule base can only be said to be correct in the sense that no test cases which produced wrong results or inconsistent results were found.

The risk of incorporating logical errors into rule bases can be significantly reduced by adopting appropriate software engineering techniques, especially during the conceptualisation of the rule base. To this end a number of expert system development environments/methodologies are available of which the best known is probably KADS ([30]). A number of formal expert system specification techniques have also been developed that result in an implementation independent (conceptual) model of the desired expert system which can be tested in isolation [15]. The perceived advantages are: that testing can take place early on in the development cycle, and that the nature of the testing is not cluttered with implementational detail [13].

Structural errors are generally detected through static inspection of the rule base. This is facilitated by the declarative logical formalism used to represent most rule bases. A great many tools and techniques have been proposed to identify errors and anomalies. These include techniques such as decision tables [28], incidence matrices [9], and rule base folding [6].

For further detail concerning error detection techniques interested readers are referred to text books such as [3] or [8], or review papers such as [29].

It is suggested here that the nature of the techniques and approaches outlined above are equally applicable to the identified overlap between database and expert systems. Some work has been done in the field of active databases to identify infinite loops (*non-termination*) using an approach known as the *trigger graph method* [1, 21]).

9. Remedial Action

Once an error or inconsistency has been detected in a rule base corrective action is required. Typically this will involve the deletion, addition or modification of one or more rules, in such a manner that the remainder of the rule base is not adversely effected. To this end it is desirable to refer to some revision strategy or plan to ensure that the corrective action is implemented in a appropriate manner, e.g. revision plans [5]. A traditional software "debugging" element will also be involved.

There are two main approaches that may be adopted to repair a flawed rule base, (1) any necessary corrective action can be implemented `by hand" or (2) software tools can be used to automatically revise the rule base. The first is a consequence of the application of software tools that simply "flag" errors and anomalies. In many cases this is considered to be all that is required. Any necessary remedial action is then implemented by the creators of the system with appropriate reference to domain experts.

Much research has been directed at the automated refinement of rule bases [11, 5], also sometimes referred to as rule base reduction [14]. The question as to what extent this is desirable (or not) is still open to debate. For example in Ginsberg's rule base reduction approach subsumed rules are automatically identified and absorbed into appropriate existing rules. However, it may be that the existence of the subsumed rule was not simply a "left over" from an earlier version of the system, but an error in logic which would then require some alternative remedial action. Consequently it is generally acknowledged that rule bases cannot be entirely revised without the eventual intervention of a domain expert, however, it is also acknowledged that automated refinement/reduction of rule bases can aid the revision process. For example the KRUST system ([12]) has, as input, a single rule base and a set of training cases. If no fault is found using these cases the original rule base is returned, otherwise one or more refined rule bases are produced which must then be evaluated.

Similar considerations are applicable to deductive and active database rule sets and clausal form relational databases. The discussion as to whether automated remedial action is desirable is also significant, and warrants further investigation.

10. Conclusions

In this paper the author has attempted firstly to identify the area of overlap between the database and expert systems communities, and secondly to present a view of the "state of the art" of rule base V&V with a view to its application to the identified overlap. Although a clear overlap between database and expert systems has been identified, there may be further areas where the experience of expert systems V&V may be applicable to database systems - this is a matter for further discussion. Given the identified overlap between the database and expert systems communities, as indicated in this paper, it is suggested that the V&V techniques established by the latter may have some application within the database community. The possible nature of this applicability is as yet unclear.

References

[1] Aiken, A., Widom, J. and Hellerstein (1992). *Behaviour of Database Production Rules: Termination, Confluence and Observable Determinism.* Proceedings ACM SIGMOD, International conference on the Management of Data, pp59-68.

[2] Abiteboul, S., Hull, R. and Vianu, V. (1995). *Foundations of Databases.* Addison-Wesley, Wokingham, England.

[3] Ayel, M. and Laurent, J.P. (1991). *Validation, Verification and Testing of Knowledge Based Systems.* John Wiley and Sons, England.

[4] Bohem, B.W. (1981). *Software Engineering Economics.* Prentice-Hall, New York.

[5] Bouali, F., Loiseau, S. and Rousset, M-C (1997). *Verification and Revision of Rule Bases.* In Hunt, J. and Miles, R. (Eds.), Research and Development in Expert System XIV, SGES publications, pp253-264.

[6] Chang, C.L., Combs, J.B. and Stachowitz, R.A. (1990). *A Report on the Expert Systems Validation Associate (EVA).* Expert Systems with Applications, Vol 1, No 3, pp219-230.

[7] Codd, E.F. (1970). *A Relational Model of Data for Large Shared Data banks.* Communications of the ACM, Vol 13, No 6, pp377-387.

[8] Coenen, F. and Bench-Capon, T. (1993). *Maintenance of Knowledge-Based Systems: Theory, Techniques and Tools.* Academic Press, London.

[9] Coenen, F.P. (1995). *An Advanced Binary Encoded Matrix Representation for Rule Base Verification.* Journal of Knowledge-Based Systems, Vol 8, No 4, pp201-210.

[10] Colmerauer, A. (1987). *Opening the Prolog III Universe: a New Generation of Prolog Promises some Powerful Capabilities.* BYTE, pp 177-182, Aug 1987.

[11] Craw, S. and Sleeman, D. (1990). *Automating the Refinement of KBS.* Proceedings ECAI' 90.

[12] Craw, S. (1996). *Refinement Complements Verification and Validation.* International Journal of Human Computer Studies, Vol 44, No 2, pp245-256.

[13] Fensel, D. (1995). *Formal Specification Languages in Knowledge and Software Engineering.* The Knowledge Engineering Review, Vol 10, No 4.

[14] Ginsberg, A. (1988). *Knowledge Base Reduction: A New Approach to Checking Knowledge Bases for Inconsistency and Redundancy.* Proceedings AAAI' 88.

[15] van Harmelen, F. and Aben, M. (1996). *Structure-preserving Specification Languages for Knowledge Based Systems.* Journal of Human Computer Studies, No 44, pp187-212.

[16] van Hentenryck, P. (1989). *Constraint satisfaction in logic Programming.* MIT Press.

[17] Jaffer, J. and Lassez, J.-L. (1987). *Constraint Logic Programming.* In Proceedings of the 14th ACM Conference on Principles of Programming Languages (POPL), Munich.

[18] Kanellakis, P.C., Kuper, G.M. and Revesz, P.Z. (1995). *Constraint query languages.* Journal of Computer System Science, Vol 51, No 1, pp26-52.

[19] Karadimce, A.P. and Urban, S.D. (1996). *Refined Trigger Graphs: A Logic-Based Approach to Termination Analysis in an Active Object-Oriented Database Management Systems.* ICDE' 96 pp384-391.

[20] Kerschberg, L. (Ed.) (1986). *Expert Database Systems: Proceedings from the First International Conference* . Benjamin-Cummings.

[21] Lee, S.Y and Ling, T.W. (1997). *Refined Termination Decision in Active Databases.* In Hameurlain, A. and Tjoa, A.M. (Eds), Database and Expert Systems Applications, (Proceedings DEXA' 97), Lecture Notes in Computer Science 1308, Springer Verlag, pp182-191.

[22] Levy, A.Y. and Rousset, M-C (1996). *Verification of Knowledge Bases Using Containment Checking.* Proceedings of AAAI' 96.

[23] Meseguer, P. (1993). *Expert System Validation Through Knowledge Base Refinement.* Proceedings IJCAI-93.

[24] Paton, N., Cooper, R., Williams, H. and Trinder, P. (1996). *Database Programming Languages* . Prentice-Hall.

[25] Preece, A.D. and Shinghal, R. (1994). *Foundations and applications of Knowledge Base Verification.* International Journal of Intelligent Systems, Vol 9, pp683-701.

[26] Robinson, J.A. (1965). *A Machine-Oriented Logic Based on the Principle of Resolution.* Journal of the ACM, Vol 12, pp23-41.

[27] Stonebraker, M., Rowe, L.A. and Hirohama, M. (1990). *The Implementation of POSTGRES.* IEEE Trans. Knowledge and Data Engineering, Vol 2, No 1, pp125-141

[28] Vanthienen, J. and Wijsen, J. (1995). *On the decomposition of Tabular Knowledge-Based System.* The New Review of Applied Expert Systems, pp77-89.

[29] Vermesan, A.I. and Bench-Capon, T. (1995). *Techniques for the Verification and Validation of Knowledge-Based Systems: A Survey Based on the Symbol/Knowledge Level Distinction.* Software Testing, Verification and Reliability, Vol 5 no 4, 1995, pp233-72.

[30] Wielinga, B.J., Schreiber, A.T. and Breuker, J.A. (1992). *KADS: A Modelling Approach to Knowledge Engineering.* Knowledge Acquisition (Special Issue: The KADS approach to knowledge engineering), Vol 4, No 1, pp5-54.

[31] Zaniolo, C. (1994). *A Unified Semantics for Active and Deductive Databases.* In Paton, N.W. and Williams, M.H. (Eds.), Rules in Database Systems, Springer-Verlag, pp271-87.