NRC Publications Archive Archives des publications du CNRC

House calls : building and maintaining a rule-base

Ruberg, K.; Cornick, S. M.; James, K. A.

This publication could be one of several versions: author's original, accepted manuscript or the publisher's version. /
La version de cette publication peut être l'une des suivantes : la version prépublication de l'auteur, la version
acceptée du manuscrit ou la version de l'éditeur.

Publisher's version / Version de l'éditeur:

Knowledge Acquisition, 1, 4, pp. 379-402, 1989

NRC Publications Record / Notice d'Archives des publications de CNRC:

<https://nrc-publications.canada.ca/eng/view/object/?id=69755865-90ae-49c0-822a-556f35493f0d>
<https://publications-cnrc.canada.ca/fra/voir/objet/?id=69755865-90ae-49c0-822a-556f35493f0d>

Access and use of this website and the material on it are subject to the Terms and Conditions set forth at

<https://nrc-publications.canada.ca/eng/copyright>

READ THESE TERMS AND CONDITIONS CAREFULLY BEFORE USING THIS WEBSITE.

L'accès à ce site Web et l'utilisation de son contenu sont assujettis aux conditions présentées dans le site

<https://publications-cnrc.canada.ca/fra/droits>

LISEZ CES CONDITIONS ATTENTIVEMENT AVANT D'UTILISER CE SITE WEB.

Questions? Contact the NRC Publications Archive team at

PublicationsArchive-ArchivesPublications@nrc-cnrc.gc.ca. If you wish to email the authors directly, please see the first page of the publication for their contact information.

Vous avez des questions? Nous pouvons vous aider. Pour communiquer directement avec un auteur, consultez la première page de la revue dans laquelle son article a été publié afin de trouver ses coordonnées. Si vous n'arrivez pas à les repérer, communiquez avec nous à PublicationsArchive-ArchivesPublications@nrc-cnrc.gc.ca.

House calls: building and maintaining a rule-base

KALEV RUBERG, STEVEN M. CORNICK AND KAY A. JAMES

Advanced Construction Technology Laboratory, Institute for Research in Construction, National Research Council Canada, 6815h, 8th St. NE, Calgary, Alberta T2E 7H7

A knowledge-acquisition system was designed and built to help an architectural firm automate their diagnosis of building problems. The system was tailored to the firm's database system and building survey method. Rules are generated from the data by the induction learning algorithms ID3 or AQ11 and the orderly development of the rule-base is ensured by a verification procedure. Architectural diagnostics rely on the expertise of an experienced analyst. Building diagnostic processes require spatial, verbal and numerical reasoning. The rigid data structure imposed by the firm excluded crucial levels of semantic information. Induction methods proved useful tools for organizing data, but the expertise was captured by *ad hoc* editing of the rule-base. This project identified a need in the building industry to develop a taxonomy and a representation to support the building diagnostic process.

1. Introduction

Induction (machine learning) algorithms are useful tools for transforming large databases into production rules. For these rules to produce meaningful inferences, the learning techniques must be augmented by less constrained methods, including manual rule authoring procedures. Rule-bases produced using manual editing facilities require verification and maintenance procedures to prevent contamination of the rule-base. A system based on these principles was developed for architects who diagnose building faults.

Numerous knowledge-based system prototypes have been developed for the building industry, most being diagnostic in nature. Typically, the prototypes focus on a particular aspect of the building. The majority are rule-based and follow a conventional pattern of knowledge acquisition; either the experts themselves built the systems or knowledge engineers construct the rule-base using various interview techniques (Davidson, Davidson & Ruberg, 1988). Such knowledge acquisition techniques limit the size of prototype systems and their extensibility.

A project undertaken by an architectural firm has emphasized the problems associated with developing a large knowledge base for building diagnosis. The ambitious nature of the project, automatic diagnosis of building problems, serve to illustrate key issues. Several experts would be needed to construct such a knowledge base and it would continue to grow as new illnesses and relationships are discovered. A systematic method of recording data and a *building related pathology* linking data to specific buildings and building illnesses would be required.

To aid in the generation of such a working knowledge base, a rule authoring assistant, designated THX 1138, was developed. The methodology is based on

incremental expansion of the knowledge base, coupled with rule-base verification at each step.

THX 1138 was designed to assist in the construction of a knowledge-based system using an architectural firm's diagnostic techniques. Rules were generated from data through the use of machine learning techniques. THX 1138 consists of three components: an interface and editor module, an inductive rule generation system (EVITA) and a rule verification system (CIRCE).

2. Diagnosing building faults

2.1. DIAGNOSTICS

Diagnostic methods range from use of causal models to diagnostic heuristic procedures based on empirical associations. Causal models are often based on simulations of physical domains and are used as comparative predictors of events. For example, a device model of a satellite's attitude controller will provide the diagnostician with an accurate representation of its performance, based on equations that describe its behaviour (Pazzani, 1986). By introducing new parameter values, errant behaviour can be replicated by a knowledgeable user. Causal simulation models are applicable in domains where the physical objects, their behaviour, and interrelationships can be described in mathematical syntax. On the other hand, domains that rely exclusively on heuristics (such as management) can be described only by observed empirical relationships. The recording and representation of these relationships is crucial to capturing knowledge and passing it on to other practitioners.

Strategies combining both causal models and heuristics are possible. These are illustrated by approaches used in diagnosing electronic components where partial system simulation models of power supplies are augmented by heuristic reasoning concerning their faults (Kahn, Kepner & Pepper, 1987). Using simulation methods to create a database of behaviour descriptions and then using induction techniques to compress the data into rules is another approach to developing heuristic diagnostic knowledge bases. Techniques for capturing both the qualitative and quantitative aspects of decision making are being explored at Xerox PARC. Harrison, (1987) has abandoned formal computer-based representations in favour of video data manipulation and recording techniques.

Within the computer-based domain acquiring the diagnostic heuristics includes:

- manual transformation of verbal reports (Hoffman, 1987) into formal algorithms or production rules,
- structured data completion and elicitation of associations (Boose, 1986)
- recording the diagnostic associations in structured databases (Michalski, Moze-tic, Hong & Lavrac, 1986)

In free verbal descriptions, formal diagnostic procedures are described as the gathering of data to support a hypothesis (forward chaining). In practice, a likely hypothesis is postulated and the facts to support it are sought. Contradictions lead the practitioner to abandon the hypothesis and embrace a new goal (commonly described as backward chaining) (Welbank, 1983). Structured elicitations use a

formal method to build and compare a table of beliefs. Boose has used the personal construct theory formulated by Kelly (1955). Structured associations are developed from a formal data representation in the problem domain—often found in database structures. The data (or knowledge) are defined *a priori*.

In order to automate the diagnostic process computationally, it is necessary to:

- have a taxonomy that supports both paradigms; one for processing (forward) and one for representing the knowledge during acquisition (backward),
- have a representation with a resolution capable of deriving the resultant causes and remedies from symptoms (or have formalisms capable of representing the entire problem space),
- have a tractable problem space.

Diagnostic procedures have been defined in the field of medicine, which provides a framework for developing pathologies based on a mix of causal and empirical associations. Within the medical domain there is agreement on taxonomy. Other fields in which diagnostic procedures are well established include VLSI and circuit board analysis. In most engineering fields the diagnostic process is constrained by the operational characteristics of the device. It is often possible to rely on causal models for diagnostic information (Tanka, 1986).

2.2. AUTOMATING THE DIAGNOSTIC PROCESS

Pioneering work in the expert system field was concentrated in medical diagnostics. Automated diagnostic systems typically specialize, for example MYCIN for blood infections (Shortliffe, 1976), NEUROLOGIST-1 for neurological disorders (Xiang, Srihari, Shapiro & Chutkow, 1984), and about 60 other systems described by Waterman (1986). Automated diagnosis is also prevalent in the VLSI industry, satellite trouble-shooting and finding faults with industrial machinery. In medical, industrial engineering, and electronics fields the diagnostic process is well constrained when compared with building related problems. Automated diagnostic systems have typically used production rule representations.

2.3. DIAGNOSTICS IN THE CONSTRUCTION FIELD

Diagnosing faults in buildings is a process similar to diagnosing medical problems. Symptomatic evidence is used to determine a cause and a related cure. There are unresolved problems, however, in developing computer-based diagnostic systems:

- no accepted taxonomy exists; loose definitions are used to describe the various parts of buildings;
- there is no “generic” building type; layout, structure, materials and operating environments are difficult to characterize and classify, (the analogy to medicine would be Dr. McCoy’s task of diagnosing the ailments of many aliens without benefit of a pathology or taxonomy);
- spatial relationships between systems are inconsistently described and not defined;
- not only are the states of objects and systems critical, but the links and the state of the relationships between systems are critical as well;

- there is no well defined pathology describing building problems;
- information about building illnesses is not organized;
- building science lacks complete simulation models on which to found causal diagnostic procedures.

Initial stages of diagnosis comprise the diagnostician's recording of symptoms and signs. For the purposes of this discussion, symptoms (reported by the user) will be combined with signs (objective physical evidence including measurements) in one category called symptoms. The symptoms are organized and a diagnosis is made. Diagnosis must be linked to the prediction of future events requiring a prognosis.

A syntax is required to relate the problem (illness) to the outward manifestations (symptoms) and their causes. A taxonomy is also needed to ensure that symptoms of problems, building-related symptoms, structures, materials, and diagnostic results, are recorded consistently. Currently building diagnosis is more of an art than a science (Wilson, 1984). Although attempts have been made to establish a building pathology for the correct diagnosis of building illnesses by Bayazit, (1984), none has led to industry standards.

2.3.1. *Manual method*

Building surveys are performed by experts with many years of field experience. Generally they rely on inspection and non-destructive testing techniques to establish a diagnosis and a prognosis. The process begins with a walk-through inspection, when the expert records his observations, makes notes, takes photographs or video tapes, and performs some non-destructive tests such as measurement of moisture content in wood measurement of crack widths or ultrasonic measurements. The diagnosis is an *ad hoc* procedure unique to each problem. Often a mental model is built that combines the condition of the component systems, the symptoms, and the related causes. Eventually all the information is deductively pieced together and the illness is identified.

2.3.2. *Automated diagnostics for buildings*

The current method of building diagnostics requires an expert practitioner, but experts are scarce and a time-consuming analysis is expensive. An architectural firm has proposed an automated method to document the information used by experienced practitioners and reduce the amount of expertise needed to perform a building diagnosis.

The walk-through method was retained but a rigid syntax was imposed on the surveyor. Alphanumeric codes describe the building systems, construction materials, and observed symptoms and link these with a cause and diagnosis. The amount of affected material is also recorded to provide remedial cost estimates automatically.

Back at the office, the expert looks at his records in the database, completes each record (symptom, cause, diagnosis) and lets the induction system identify patterns in the data. The expert then checks the rules, adds new attributes if he thinks a meta-level is required and verifies the manually modified rules with an automated consistency checker.

Rules developed from the database are posted to the general rule-base. When the next database describing a building is given to the expert for analysis, he invokes a

knowledge based system (using the generated rule-base from previous diagnoses) to draw as many conclusions as possible from the existing rule-base. He then looks for patterns and generates rules from the unused data and repeats the previous iteration.

Rules in the system are written in a specific syntax unique to the knowledge acquisition and verification module. The syntax is translated to conform to one of three knowledge-based systems shells used to analyse incoming data. This automated diagnosis procedure and THX 1138 are shown in Fig. 1.

2.3.3. Knowledge representation

A first attempt by the architectural firm to automate the diagnostic process used a rigid syntax to describe the building problem. Information at the site was captured by a novice architect who referred to a guide for the required syntax. The findings were recorded on Dictaphone tapes. These were transcribed at the office into alphanumeric codes representing the material and symptom descriptions. The site was described by taking a prescribed path around and through the building. As the surveyor gained more experience, the use of the alphanumeric codes in place of actual words became more common.

For each record, the expert included a cause (and a solution) in the description of an individual problem. The editor facilities for completing the problem description record were, however, inadequate and difficult to use. The alphanumeric syntax stripped contextual, spatial and ancilliary information from the problem description. For building diagnosis, the data in the file no longer represented the actual problem at the site.

In addition to the alphanumeric coded description of the building, a free-form "natural language" description was permitted in the "memo" part of the database. An example description of a concrete building is presented in Fig. 2. Although most of the software development effort for this diagnostic representation was found in the alphanumeric code of the database, it was the memo field that provided the expert with the basis for diagnosing the surveyed building illnesses.

The data structure described problems in a building according to general location (zone by floor), material involved (e.g. brick or concrete using a material specification code that is well accepted by the building design community), system in a building (e.g. wall system, mechanical system), symptom exhibited, and a prognosis for each problem. Each record was assumed to contain a complete problem description. As all the fields were recoded as alphanumeric codes, it was difficult for the expert to read the descriptive evidence presented in each record. An example from a file is shown in Fig. 3. (By the end of this project, one diagnostician in the firm was quite capable of reading these alphanumeric codes and describing the contents of each record.)

2.3.4. Knowledge acquisition

The task of building such a "general practitioner" cannot be handled by informal knowledge acquisition methods. An estimate of the number of medium-grained rules in the knowledge base for a competent building ailment diagnostic system (for northern latitudes of North America) exceeded 100 000. This estimate is based on a diagnostic system for window problems. (It contains about 200 rules for diagnosing

FIG. 1. From the manual survey (A) the information is transcribed (B) into alphanumeric codes in the database (C). The knowledge-based system (KBS) uses forward chaining rules (D) to produce a fault report (E). THX1138 uses the database information to create the forward chaining rules using induction (1). These induced rules are posted to an internal rule-base (2) and edited (3) using an EMACS style editor. The internal rule base is verified (4) and translated (5) into the rule-syntax of the KBS shell.

Zone: third floor, facade, south-west orientation.
 Memo: facade has a crack running from the third floor to the second floor
 System code: SS0302000 (locked up in field)
 Memo: stucco on brick facade
 stucco is cracked and spalling
 about 80 years old
 ...

This facade problem would be transcribed as:

Zone: third floor, facade, south-west orientation.
 Memo: facade has a crack running from the third floor to the second floor
 Systeme code: SS0302000
 Memo: stucco on brick facade
 Age: 1910
 Maintenance: 9 (high)
 Use: 0 (not operable)
 Placement: Part of wall
 Exposure: Normal
 Material code: MT970200 ...

FIG. 2. Formal of information developed at the survey site and its transcription at the office. During the second stage, a CAUSE and ILLNESS are associated with the record.

one of the five ailments in the component. A complete diagnostician for the component might contain 1000 rules and there are about 50 discrete large-scale components with two systems-description levels in most buildings. This estimate does not account for the required metal level description of the problem space.)

Welbank (1983) has described three phases typically followed in knowledge acquisition for knowledge-based system development. The first phase comprises knowledge representation or basic data structure; the second, production of a working system (by the elicitation of knowledge, verification of rule semantics and syntax); and the third, validation or testing and debugging to achieve a level of confidence in the system. The reported computer-based modules built for the

(NIL NIL MT099100010 NIL SP030200000 NIL NIL NIL 1967 A 5 NIL NIL NIL NIL 7 NIL NIL NIL NIL NIL 10.0 NIL NIL 1 3 CA020400000)
 (NIL NIL MT085100010 NIL SP040600000 NIL NIL NIL 1922 R 7 NIL NIL NIL NIL NIL 9 NIL 180 NIL NIL 100.0 NIL NIL 5 5 CA020400000)
 (NIL NIL MT085100010 NIL SP020110000 NIL NIL NIL 1922 R 7 NIL NIL NIL NIL NIL NIL NIL NIL NIL 100.0 NIL NIL 5 8 CA020400000)
 (NIL NIL MT081100000 NIL SP030200000 NIL NIL NIL 0 A 9 NIL NIL NIL NIL NIL NIL NIL NIL NIL 100.0 NIL NIL 1 5 CA000000000)
 (NIL NIL MT057000010 NIL SP040600000 N NIL NIL NIL 1922 R 7 NIL NIL NIL NIL NIL NIL 5 NIL NIL NIL NIL 5.0 NIL NIL 5 3 CA020400000)
 (NIL NIL MT079200000 NIL SP030200000 NIL NIL NIL 0 1922 A 7 NIL NIL NIL NIL NIL 7 NIL NIL NIL NIL NIL 100.0 NIL NIL 5 9 CA020400000)
 (NIL NIL MT057000010 NIL SP040300000 N NIL NIL NIL 1922 R 7 NIL NIL NIL NIL NIL NIL 5 NIL NIL NIL NIL 3.0 NIL NIL 1 0 CA020400000)
 (NIL NIL MT034100010 NIL SP020107000 N NIL NIL NIL 1922 R 7 NIL NIL NIL NIL NIL NIL 5 9 NIL 180 NIL NIL 50.0 NIL NIL 5 5 CA010304000)
 (NIL NIL MT042100000 NIL SP060100000 O NIL NIL NIL 1922 A 5 NIL NIL NIL NIL NIL NIL 5 9 NIL 180 NIL NIL 100.0 NIL NIL 7 0 CA010304000)
 (NIL NIL MT042200000 NIL SP060100000 O NIL NIL NIL 1922 A 5 NIL NIL NIL NIL NIL NIL 5 9 NIL 180 NIL NIL 2+.0 0.0 NIL 7 0 CA010304000)
 (NIL NIL MT033100010 NIL SP020107000 NIL NIL NIL 1922 R 7 NIL NIL NIL NIL NIL NIL 5 9 NIL 180 NIL NIL 2.0 0.0 NIL 5 5 CA030000000)
 (NIL NIL MT033100010 NIL SP010200010 N NIL NIL NIL 1922 R 7 NIL NIL NIL NIL NIL NIL 5 9 NIL 180 NIL NIL 2.0 NIL NIL 5 2 CA010305020)
 (NIL NIL MT042200000 NIL SP010200010 N NIL NIL NIL 1922 I 9 NIL NIL NIL NIL NIL NIL 5 9 NIL 360 NIL NIL 1.0 NIL NIL 7 9 CA010305020)
 (NIL NIL MT092200000 NIL SP020107000 NIL NIL NIL 1967 A 9 NIL NIL NIL NIL NIL NIL 9 NIL 360 NIL NIL 32.0 NIL NIL 5 7 CA010305020)
 (NIL NIL MT099100010 NIL SP040600000 NIL NIL NIL 1967 A 5 NIL NIL NIL NIL NIL NIL 5 9 NIL 360 NIL NIL 5.0 NIL NIL 1 3 CA010300000)
 (NIL NIL MT099100010 NIL SP030200000 NIL NIL NIL 1967 A 5 NIL NIL NIL NIL NIL NIL 9 NIL 360 NIL NIL 10.0 NIL NIL 1 3 CA010301060)
 (NIL NIL MT085100010 NIL SP040600000 NIL NIL NIL 1922 R 7 NIL NIL NIL NIL NIL NIL 9 NIL 360 NIL NIL 100.0 NIL NIL 5 5 CA020400000)

FIG. 3. Database records describing the problems in a concrete building. These are developed from the transcripts of site surveys and use an alphanumeric code to describe the material, structure, symptom, cause and prognosis. For example, alphanumeric codes beginning with MT indicate materials; codes with SP indicate symptoms. The fields without any information (NIL) comprise more than half the contents of the database.

architectural firm focus on the second phase of knowledge-based system building—that of eliciting the knowledge and structuring it in a mode compatible with a chosen representation. Development was limited by the pre-existing data structure and data representation.

3. Applied machine induction techniques

The architectural firm's survey methodology produced many data. About 500 database records formed a hierarchical description of a surveyed building. The underlying assumptions leading to the use of machine learning algorithms to extract the diagnostic knowledge were completeness of each record and a structured pathology that included the expertise involved in diagnosing building problems. The availability and structure of the information suggested the use of a combination of induction techniques, namely learning from examples and learning by observation and discovery.

In the process of learning from examples, a set of examples is provided by a teacher. The example set may or may not contain descriptions of concepts. In one scenario the teacher may generate an example set that is meant to be as helpful as possible; on the other hand, a database file from a field survey may contain relatively uncontrolled observations. In the present case only positive examples were provided to the learner, whose goal was to classify observations of multiple aspects of the environment with minimum interaction with the teacher.

3.1. EVITA

An incremental learning approach was chosen for EVITA. (EVITA was developed from an earlier version of software named KATS). The learner must form one or more hypotheses consistent with the available data and subsequently refine them after considering additional examples.

3.2. COMPARISON OF LEARNING ALGORITHMS

Two machine learning algorithms were used to construct the rule authoring assistant EVITA, specifically a version of Quinlan's (1986) Iterative Dichotomizer 3 (ID3) algorithm and Michalski's (1983a) AQ11 program. The ID3 variant builds decision trees to discriminate among classes of objects in the domain. Nodes of the decision tree correspond to selected object attributes such as symptom, material, and orientation. Edges correspond to predetermined alternative values for these attributes. The leaves of the tree correspond to sets of objects with an identical classification. Decisions as to what attributes are appropriate for branching are made on statistically derived characteristics of the example set, e.g. an entropy type statistic based on information theory or calculation of the chi-squared ratio.

The AQ11 algorithm is typical of the generalization-to-specialization concept of machine learning. Generalization begins with the most general possible hypothesis and moves towards more specific descriptions. When a hypothesis fails to cover a positive occurrence it is deemed overly specific and more general hypothesis is formulated. If a hypothesis covers a negative instance it is judged to be overly general and subsequently dismissed. The inverse of generalization is specialization; if a hypothesis is deemed to be overly general (by covering negative instances) it is

replaced by a more specific hypothesis. Hypotheses that do not cover all the positive examples are dismissed as overly specific. This type of induction is heuristic in nature, using specialization and generalization rules (Michalski, 1983*b*) as opposed to strategies such as ID3 that are statistical in nature.

It was possible to compare the performance of both induction algorithms using the database files provided by the architectural firm. Each algorithm has a distinct representation; ID3 represents knowledge as a classification tree, while this particular AQ11 implementation represents knowledge as discrete events and hypotheses using formal logic-based expressions. Both representations were converted to production rules for use by the architectural firm in their knowledge-based system. Classification trees from both ID3 and AQ11 have been derived from the datafile and are shown in Fig. 4.

If the complexity of a rule is measured by the number of antecedent clauses and attributes in a rule, then ID3 tended to produce more complex rules than AQ11. Conversely, the rules generated by AQ11 were easier to understand. The explanation for this lies with the decision tree representation of ID3. The construction of decision trees occurs in a depth-first fashion. Each complex rule generated by ID3 included all the previous attributes in the decision tree.

AQ11 creates rules from similar data more slowly than ID3, which demonstrates a linear response with increasing problem size. AQ11 exhibits exponential behavior. Although ID3 is a one-step induction algorithm, it was possible to achieve some level of incremental learning with it, but this involved considerable user interaction.

A superior feature of AQ11 is the flexibility accorded the expert in incorporating rules of thumb and domain-specific information into the induction process. Domain information in the ID3 methodology must be incorporated as attributes, attribute values, and class values. The data file in Fig. 3 contains the information of a site survey carried out on a concrete building. ID3 was used to classify this data file, but when the rules were reviewed the experts commented that it had generated incomplete rules. Instances of "building problem causes" lacked an attribute specifying the material. This implied that the material was irrelevant to the building problem being diagnosed. The generated rule did not embody the expert's opinion of the material's importance. ID3 was unable to discriminate between certain class values of illnesses using the attribute "material." Some domain knowledge such as materials and illnesses must be related, or some domain facts such as "the building is made of concrete" would have to be included to ensure that rules are generated in the proper context.

The information contained in building surveys can be numerical as well as categorical. Information such as material quantities, percentage of areas affected, and age of building and materials is recorded. The ID3 algorithm was extended to allow for the inclusion of numerical attributes. These are converted into partitions, which are represented as Boolean attributes. One method of partitioning a numerical attribute space is to break the space at the mod-points. For example, given the following data:

building age = 10, 20, 40

the space would be partitioned into the following Boolean attributes:

building age < 30, building age < 15

```

(CODMAT(MT075100000(CAUSES(CA020400000)))(MT075400000(CAUSES(CA10301060)))
(MT099100010
(CODSYM
(SP030200000
(ORIENT(NIL(CAUSES(CA020400000)))(360(CAUSES(CA010301060)))
(90(CAUSES(CA020400000)))))
(SP020107000(CAUSES(CA020400000)))
(SP040600000
(ORIENT(360(CAUSES(CA010300000)))(270(CAUSES(CA010206000)))
(90(CAUSES(CA010206000)))(180(CAUSES(CA010206000)))))
(MT084100010(CAUSES(CA020400000)))
(MT033100010
(STADE
(5
(ORIENT(360(CAUSES(CA010305020)))(270(CAUSES(CA010305020)))
(180(CAUSES(CA030000000)))))
(3(CAUSES(CA030000000)))(2(CAUSES(CA010305020)))))
(MT092200000(CAUSES(CA010305020)))(MT085100010(CAUSES(CA020400000)))
(MT057000010(CAUSES(CA020400000)))(MT034100010(CAUSES(CA010304000)))
(MT079200000(CAUSES(CA020400000)))(MT042100000(CAUSES(CA10304000)))
(MT042200000(STADE(9(CAUSES(CA010305020)))(0(CAUSES(CA010304000)))))
(MT081100000(STADE(5(CAUSES(CA000000000)))(3(CAUSES(CA020400000)))))
(MT057000000(CAUSES(CA020400000)))(MT079200010(CAUSES(CA020104000)))
(MT057100010(CAUSES(CA020400000)))(MT082100010(CAUSES(CA020400000)))
(MT042000000(CAUSES(CA000000000)))))

```

(a)

HYPOTHESES

```

(((STADE = 8)V((CARACT = 5)(STADE = 9))V((CARACT = 5)(CODSYM = SP040600000))V
(CODMAT = MT057000010)V((CODSYM = SP030200000)(ORIENT = 90))V
((ORIENT = NIL)(STADE = 3))V((CARACT = 1)(STADE = 0))V
((CODMAT = MT099100010)(CODSYM = SP020107000))V((ORIENT = 360)(CODMAT =
MT081100000))
V(CODMAT = MT082100010))
-(CAUSES = CA020400000))
(((CODMAT = MT075400000)V((CODSYM = SP030200000)(CODMAT =
MT099100010)(ORIENT = 360))
-(CAUSES = CA010301060))
(((CODMAT = MT033100010)(ORIENT = 270))V(CODMAT = MT092200000)V(CODSYM =
SP010200010)
V((CODMAT = MT033100010)(ORIENT = 360)))
-(CAUSES = CA010305020))
(((CODSYM = SP040600000)(CARACT = 1)(ORIENT = 270))V
((CODSYM = SP040600000)(CARACT = 1)(ORIENT = 90))V((CARACT = 1)(ORIENT = 180)))
-(CAUSES = CA010206000))
(((CODMAT = MT034100010)V(CODSYM = SP060100000))-(CAUSES = CA010304000))
(((CODMAT = MT033100010)(STADE = 3))V((CODMAT = MT033100010)(ORIENT =
180)(STADE = 5))
-(CAUSES = CA030000000))
(((CODMAT = MT081100000)(STADE = 5))V(CODMAT = MT042000000))-(CAUSES =
CA000000000))
(((CODSYM = SP040600000)(CARACT = 1)(ORIENT = 360))-(CAUSES = CA010300000))
((CODMAT = MT079200010)-(CAUSES = CA020104000)))

```

(b)

Fig. 4. A comparison of the generated classifications for the same data using ID3 and AQ11. (a) an AD3 decision tree; (b) an AQ11 rule.

```

(HYPOTHESES
((((CODZON = NIL) (CODSYS = NIL) (CODMAT = MT099100010) (SYSMAT = NIL) (CODSYM = SP040600000)
(EMPLAC = NIL) (PIECE = NIL) (ETAGE1 = NIL) (ETAGE2 = NIL) (AGEMAT = 1967) (INTMAT = A)
(ENTMAT = 5) (USAMAT = NIL) (QUAMAT = NIL) (UNIMAT = NIL) (AGESYS = NIL) (INTSYS = NIL)
(ENTSYS = NIL) (USASYS = NIL) (MILIEU = 5) (EXPOSI = 9) (FACADE = NIL) (ORIENT = 270)
(SUPERF = NIL) (VALEUR = NIL) (AFFECT = NIL) (QUASYM = NIL) (UNISYM = NIL) (CARACT = 1)
(STADE = 3))
V
((CODZON = NIL) (CODSYS = NIL) (CODMAT = MT099100010) (SYSMAT = NIL) (CODSYM = SP040600000)
(EMPLAC = NIL) (PIECE = NIL) (ETAGE1 = NIL) (ETAGE2 = NIL) (AGEMAT = 1967) (INTMAT = A)
(ENTMAT = 5) (USAMAT = NIL) (QUAMAT = NIL) (UNIMAT = NIL) (AGESYS = NIL) (INTSYS = NIL)
(ENTSYS = NIL) (USASYS = NIL) (MILIEU = NIL) (EXPOSI = 9) (FACADE = NIL) (ORIENT = 90)
(SUPERF = NIL) (VALEUR = NIL) (AFFECT = 2.0) (QUASYM = NIL) (UNISYM = NIL) (CARACT = 1)
(STADE = 3))
V
((CODZON...

```

FIG. 5. Internal rule syntax used by the modules and editor before translation to a specific KBS shell format.

Several other partitioning heuristics such as partitioning around the mean were provided, as well the ability to include other heuristics as needed. AQ11 treats numbers as categorical values; it generated some hypotheses that were difficult to follow owing to the preponderance of numerically dependent antecedents.

The data supplied by the architectural firm consisted of a relatively square database having approximately 30 attributes for 60 record entries. Approximately half of the records were empty. Neither induction method handled missing data. Rules with missing attribute values were in the majority. A portion of a generated rule is shown in Fig. 5. Over one half of the attribute values are NIL. The rule contains clauses with up to 30 attributes. When the attributes containing no information are excluded from the induction process, the rules become more understandable. In fact, after several trial runs with this database it was found that only 3 of 30 attributes; symptom, material, and orientation, proved necessary to classify the building illnesses.

3.3. MANUAL EDITING OF RULE-BASES

The experts wished to include additional levels of attributes in the generated rules in order to arrive at meaningful descriptions of illnesses. This could have been achieved by providing negative or counter-examples in the data thereby forcing the induction algorithms to include certain parameters. It was found easier, however, to add antecedent clauses to specific rules. In the example generated from the concrete building, the additional attribute "MATERIAUX = BETON" was added to some of the rule clauses.

Allowing the experts to change and develop their own rules introduced a new set of problems:

- (1) the rules were no longer necessarily logical or consistent
- (2) the edited rules no longer precisely reflected the data recorded on site
- (3) the semantic consistency changed with addition of new attributes (to define meta-level causes and diagnoses)

Only the first of these concerns was effectively dealt with. A rule verification algorithm was developed to maintain logical consistency in the rule-base. The rule

editing facility in EVITA was a simple EMACS type text editor and provided no help to the user.

The weakest link in this implementation of a rule-base authoring method proved to be the manual editing facility. It relied on an EMACS type editor and displayed rules to the expert in the format shown in Fig. 5. This format and the EMACS text interface were not conducive to freely entering associations. On the other hand, it did provide the rule author (once familiar with the system) a much greater latitude in developing new OAV associations.

The same editor was used to change rules identified by batch runs of the verification algorithm. A separate interface for the verification task needs to be developed.

4. Keeping the record straight: Rule-base Verification

The automatic analysis of a database by means of an induction algorithm will not give rise to a logically inconsistent rule set. When the rules produced by multiple applications of the algorithm, using different combinations of parameters, are merged, then the resulting rule-base may contain examples of redundancy and/or subsummation. As rules derived from additional databases are added to the knowledge-base, and rules are added or modified manually by the experts, the danger of compromising the logical integrity of the system increases.

In order to maintain the logical consistency of the rule-base, it was necessary to adopt explicit verification methodologies, and to implement them in such a way that they could be used during development, after each modification or addition. Verification is a process supplementary to validation, that tests the performance of a system and can therefore only be used when the system, or a prototype, is largely complete. Validation techniques treat the system as a "black box", comparing the conclusions arrived at by the system with those of the expert, given the same initial data. When there is failure to correlate, adjustments are made to the system. Generally, a system is developed with reference to an example set of data and solutions, extended to cope with extreme cases, and tested on new data. Although it is an essential part of acceptance testing, this process cannot constitute a formal proof since it is inductive; however comprehensive the testing there is no guarantee that further, unconsidered circumstances will not prove disastrous. Moreover, confidence in the reliability of a system requires a more solid base than assurances that it has worked so far. Not only must the system be correct, it must be seen to be correct. To achieve this level of confidence, it is necessary to verify the internal logic of the knowledge-base and inference engine (Suwa Scott & Shortliffe, 1982).

In a rule-based system there are several possible causes of error or less than optimal performance:

- (1) Rules may be in conflict.
- (2) Rules may be redundant.
- (3) Inference chains may be circular.
- (4) Rules, inference chains or attributes may be isolated, either never invoked by the rest of the system or not capable of fulfilment.
- (5) Attributes may be incomplete, having some range of their value unreachable (Nguyen, Perkins, Laffey & Pecora, 1985).

Where the rule-base is small and developed by one person over a limited period of time, such faults can usually be discovered fairly readily by inspection. In more complex situations (the building diagnostics area is a prime example) the process of assessing logical integrity is non-trivial. Problems arise when the knowledge is fragmented and contributions from several experts are to be incorporated. Experts may suggest rules that contradict, conflict with or overlap rules from other experts; they may use different terms for the same concept or the same term for different concepts, and the root of these differences may be conceptual or merely syntactic. Such problems may also occur when new knowledge is added to an existing rule-base, particularly where the additions are not made by the original knowledge engineer.

In these situations an automatic verification system is useful, not only to increase the level of confidence in the completed rule-base but also as a tool during development. Since a knowledge-base is essentially a logical system, it is amenable to logic-based verification techniques (Stachowitz, Coombs & Chang, 1987).

4.1. RULE-BASE LOGIC

In order to evaluate the logical dependencies in a rule-base, it is necessary first to examine the structure of a single rule. The antecedent of the rule may consist of one or more "literals", simple assertions of the type "attribute A has value B" or " $A = B$ ". These literals may be linked conjunctively or disjunctively, i.e. by the Booleans "and" or "or". Conjunctive literals are each necessary but not sufficient for the rule to fire; disjunctive literals are sufficient but not necessary. A disjunctive clause may consist of one or more conjunctive literals. It follows that a literal not conjoined with another literal is either a complete or at least a sufficient antecedent for the rule. When comparing rules, it is necessary to compare not only complete antecedents but also sufficient antecedent clauses. The consequent clause of the rule is generally a single literal; rules with multiple consequents may be considered as a set of separate rules with identical antecedents.

The following types of logical inconsistency in a rule-base have been identified;

4.1.1. *Conflict*

When the same antecedent (or sufficient antecedent clause) occurs in two rules whose consequents are contradictory, there is obviously a problem—one rule at least is incorrect:

```
if  A is B or C is D
then P is Q
```

```
if  A is B
then P is not Q
```

Where the consequents reach different conclusions about the same attribute, the issue is context-dependent:

```
if  sky is cloudy
then temperature is low
```

if sky is cloudy
then temperature is high

represents a conflict, whereas

if sky is cloudy
then temperature is low

if sky is cloudy
then temperature is likely to change

does not.

Rarely, a single rule may be self-contradictory:

if A is B **or** C is D **or** E is F
then C is not D

4.1.2. Redundancy

There are several ways in which rules can overlap. While not usually a fatal error, this can lead to inefficiency, to undue weight being given to repeated information, or can mask mistaken omissions. A rule may be redundant:

if A is B **or** C is D
then P is Q

if C is D **or** A is B
then P is Q

Where two rules reach the same conclusion and have equivalent antecedents or antecedent clauses, then either one rule is subsumed by the other or parts are redundant.

if A is B **or** C is D
then P is Q

if A is B
then P is Q

first rule subsumes second

if A is B
then P is Q

if A is B **and** C is D
then P is Q

first rule subsumes second

if A is B **or** C is D **and** E is F
then P is Q

if C is D **and** E is F **or** G is H
then P is Q

one occurrence of the condition “ C is D and E is F ” is redundant.
Note that

if A is B **and** C is D
then P is Q

if C is D **and** E is F
then P is Q

is not an example of redundancy; although it would be possible to combine the rules, the resulting disjunction would be logically equivalent to the two separate rules.

Where the antecedents are almost equivalent but contain contradictory literals, yet reach the same conclusion, that literal is an irrelevant condition:

if A is B **and** C is D **and** E is F
then P is Q

if A is B **and** C is D **and** E is not F
then P is Q

“ P is Q ” does not depend on the truth of “ E is F ”.

4.1.3. *Circularity*

When a rule refers to itself, either directly or indirectly, then on firing that rule the system will hang.

if A **and** B **or** C
then B self-reference

if A **and** B **or** C
then P

if P **or** Q
then X

if X
then C circular inference chain.

4.1.4. *Incompleteness*

Rules whose conclusions are not accessed by other rules, and do not constitute goal-states, represent “dead-ends” in an inference chain. Rules with antecedent conditions that are not capable of fulfilment (from initial or user-interface input, or as a consequent of another rule) cannot fire. Incompleteness of chains is flagged.

An attribute may be incomplete, having some part of its allowable range of values not covered by any rule. This check was not implemented in the verification module.

4.2. CIRCE

The verification system CIRCE (Conflict, Incompleteness, Redundancy and Circularity Evaluator) was developed as a knowledge acquisition tool for the building diagnosis package. It was written originally on a Lisp machine and then ported to an

AT class microcomputer running Common Lisp for integration with the classification module. CIRCE examines the rule-base, checking each rule for internal problems, then comparing it with every other rule in turn and noting interaction problems and dependencies. Finally, each inference path is checked for faults.

The system first sets up empty stacks for the list of rules and the reports of faults. It then calls, serially, five sub-modules, performing the necessary procedures of reading in the rule-file, separating out the logical components of the rules, comparing the rules and charting the logical dependencies, investigating inference chains, and reporting the results.

4.2.1. Reading the rule-file

The first sub-module opens the specified file and passes on each rule in turn, in the form of a list. The first item in each list (which is the name of the rule) is added to the global variable "rulestack" and the name and text are passed on. The data structure used for the rules is the "frame" (Winston & Horn, 1984). Frames can have an unlimited number of slots, each of which can have an unlimited number of facets with value assigned to each. A frame is set up for each rule, referenced by the rule-name. The antecedent and consequent portions of the text are separated and placed on the frame.

4.2.2. Parsing the rules

The parsing submodule analyses the rules for logical comparison. First, in cases where the consequent contains a negative, "not" is placed at the beginning of the text. Next, the text of the antecedent is divided, by splitting on "or", into separate disjunctive parts; these are further divided, by splitting on "and", into separate conjunctive clauses, which are recombined after alphabetical sorting. Each clause is given a unique reference.

Note that a disjunctive clause may contain one or more conjunctive clauses, and is always a sufficient condition for the rule. An antecedent may contain one or more disjunctive clauses. A rule whose antecedent consists of multiple disjunctive clauses is logically equivalent to that number of separate rules, each with the same consequent. A conjunctive clause is always a single statement, a "literal", but may also be disjunctive, i.e. constitute a complete antecedent.

4.2.3. Checking logical dependencies

The next submodule checks the logical relationships between rules. Each rule in turn is first examined for internal problems, (self-contradiction and self-reference), then compared with every other rule on the remainder of the list of rules. In this way, each rule is checked only against those rules with which it has not yet been compared.

Rules with equivalent antecedents or antecedent clauses are checked for several possible errors. If the rule consequents are contradictory (one is the Boolean complement of the other) there is contradiction; if they refer to the same parameter but reach different conclusions there may be conflict, depending on the type of parameter. As no information about parameters is available in this implementation, the situation is simply presented to the user and the parameter and the rules concerned are posted to the appropriate variable.

If the rule consequents are the same, there is repetition, which can lead to inefficiencies and inaccuracy in the rule-base since the same information fires two rules. When the rules are identical, the situation is reported as redundancy; when there are alternative antecedents the less constrained rule is described as subsuming the more constrained (i.e. the rule with fewer alternatives). If the consequents are

;;; TEST-RULES FOR ERROR CHECKING

;; This rule-base contains at least one example of each type of error discoverable by ;; the experimental error-checking program "Circe".

;SECTION 2 - rules

(rule1 wall-structure is metal stud ^ rainscreen is brick - wall is prone to cracking)
 (rule2 air-barrier is continuous - wall is dry)
 (rule3 wall-structure is concrete brick ^ rainscreen is missing V wall is cavity construction - wall is prone to leakage)
 (rule4 wall-structure is metal stud ^ rainscreen is brick - wall is not prone to cracking)
 (rule5 rainscreen is brick ^ wall-structure is metal stud - wall is prone to wind penetration)
 (rule6 wall is cavity construction V wall-structure is concrete brick ^ rainscreen is missing - wall is prone to leakage) (rule7 curtain-wall is not glazed with metal mullions - curtain-wall is structural glazing)
 (rule8 wall is seamless glass V curtain-wall is not glazed with metal mullions - curtain-wall is structural glazing)
 (rule9 wall is dry V wall is seamless glass ^ curtain-wall is not glazed with metal mullions - wall-performance is good)
 (rule10 curtain-wall is structural glazing V wall is seamless glass ^ curtain-wall is glazed with metal mullions - wall-performance is good)
 (rule11 wall is dry - air-barrier is continuous)
 (rule12 wall-structure is concrete brick ^ rainscreen is missing V wall is cavity construction - rainscreen is not missing)
 (rule13 wall-structure is concrete brick ^ rainscreen is missing V wall is cavity construction - wall is cavity construction)
 (rule14 roof is membrane - roof is to be covered with reflective aggregate)
 (rule15 wall-structure is concrete brick ^ rainscreen is not missing V wall is cavity construction - wall is prone to leakage)
 (rule16 wall is cavity construction ^ air-barrier is not continuous V wall-structure is concrete brick ^ rainscreen is missing - wall is prone to leakage)
 (rule17 roof is not membrane - roof is to be covered with reflective aggregate)

RESULTS of CIRCE'S evaluation of the rule-base.

The following logical inconsistencies were discovered:-

RULE13 is self-referential. RULE13: IF (WALL-STRUCTURE IS CONCRETE BRICK ^ RAINSCREEN IS MISSING V WALL IS CAVITY CONSTRUCTION) THEN (WALL IS CAVITY CONSTRUCTION)

RULE12 is self-contradictory. RULE12: IF (WALL-STRUCTURE IS CONCRETE BRICK ^ RAINSCREEN IS MISSING V WALL IS CAVITY CONSTRUCTION) THEN (NOT RAINSCREEN IS MISSING)

Rules RULE1 and RULE4 are contradictory. RULE1: IF (WALL-STRUCTURE IS METAL STUD ^ RAINSCREEN IS BRICK) THEN (WALL IS PRONE TO CRACKING) RULE4: IF (WALL-STRUCTURE IS METAL STUD ^ RAINSCREEN IS BRICK) THEN (NOT WALL IS PRONE TO CRACKING)

Rules RULE1 and RULE21 are contradictory. RULE1: IF (WALL-STRUCTURE IS METAL STUD ^ RAINSCREEN IS BRICK) THEN (WALL IS PRONE TO CRACKING) RULE21: IF (WALL-STRUCTURE IS METAL STUD ^ RAINSCREEN IS BRICK V WALL IS DRY) THEN (NOT WALL IS PRONE TO CRACKING)

There is redundancy between rules RULE3 and RULE6 RULE3: IF (WALL-STRUCTURE IS CONCRETE BRICK ^ RAINSCREEN IS MISSING V WALL IS CAVITY CONSTRUCTION) THEN (WALL IS PRONE TO LEAKAGE) RULE6: IF (WALL IS CAVITY CONSTRUCTION V WALL-STRUCTURE IS CONCRETE BRICK ^ RAINSCREEN IS MISSING) THEN (WALL IS PRONE TO LEAKAGE)

There is redundancy between rules RULE3 and RULE15 RULE3: IF (WALL-STRUCTURE IS CONCRETE BRICK ^ RAINSCREEN IS MISSING V WALL IS CAVITY CONSTRUCTION) THEN (WALL IS PRONE TO LEAKAGE) RULE15: IF (WALL-STRUCTURE IS CONCRETE BRICK ^ RAINSCREEN IS NOT MISSING V WALL IS CAVITY CONSTRUCTION) THEN (WALL IS PRONE TO LEAKAGE)

There is redundancy between rules RULE6 and RULE15 RULE6: IF (WALL IS CAVITY CONSTRUCTION V WALL-STRUCTURE IS CONCRETE BRICK ^ RAINSCREEN IS MISSING) THEN (WALL IS PRONE TO LEAKAGE) RULE15: IF (WALL-STRUCTURE IS CONCRETE BRICK ^ RAINSCREEN IS NOT MISSING V WALL IS CAVITY CONSTRUCTION) THEN (WALL IS PRONE TO LEAKAGE)

There is redundancy between rules RULE3 and RULE16 RULE3: IF (WALL-STRUCTURE IS CONCRETE BRICK ^ RAINSCREEN IS MISSING V WALL IS CAVITY CONSTRUCTION) THEN (WALL IS PRONE TO LEAKAGE) RULE16: IF (WALL IS CAVITY CONSTRUCTION ^ AIR-BARRIER IS NOT CONTINUOUS V WALL-STRUCTURE IS CONCRETE BRICK ^ RAINSCREEN IS MISSING) THEN (WALL IS PRONE TO LEAKAGE)

Fig. 6. Generated rule-base fault report from the verification program. These faults were generated from the working rule-base listed in the box above the report.

the same and the rules have contradictory antecedents, or complex antecedents that are the same except for a single literal asserted in one and denied in the other, that condition is irrelevant to the truth of the consequent.

Finally, information on the dependencies between rules is compiled in preparation for the next stage.

4.2.4. Examining inference paths

The information recorded while searching the rule-base is now used to examine all possible inference paths. Starting from each rule in turn, those rules for which the current rule's consequent is a condition are checked; if the consequent of the rule examined is a literal in the antecedent of the original rule, there is circularity. The descendants of each of the rules are also checked, enabling circularity to be detected however long the inference chain. The search is depth-first.

4.2.5. Reporting

When the verification procedures are completed and no faults have been reported, a message is sent to the screen. If there are faults, the appropriate report-printing function is called and the material recorded for each instance of the fault. A file is opened for the results, and for each fault appropriate explanatory text and the text of the rules is printed. An example of a rule-base and the identified errors is shown in Fig. 6.

4.3. USE OF CIRCE

After editing, the verification procedure should be re-called until no faults are reported. Search time depends on the number of comparisons to be made (which is less than the number of rules squared). Comparisons take minimal time if there are no similarities between rules, so that the total time needed depends on the complexity of the rule-base; a rule-base of fewer than 50 rules, designed to incorporate an example of every fault-type described above, was processed in under 3 min on an AT class machine. This short processing time should encourage regular use of the verification procedure, whenever modifications of any substance have been made to the rule-base.

5. Diagnosis

The system has been used in-situ on relatively small files (two files of 60 records) due to the unavailability of complete records. The rules generated by the induction algorithms were perceived by the experts to be "obvious" and "trivial". On the other hand, even with the small data sample, the architects discovered relationships and patterns they had not considered before.

Verification of other diagnostic rule-bases, such as those found in a rule-base for diagnosing problems with windows, found two redundancies among 195 rules and brought 18 potential conflicts to the author's attention, three of which were other than allowable multiple values. In all, five fatal faults were identified.

This system was envisioned as underpinning a long-term effort by the firm to develop the knowledge-base. The tool has actually pointed to weakness in the computer representation of building faults.

5.1. REPRESENTATION

During the data representation development phase the firm had assumed that experts were using data in the records to arrive at diagnoses when in practice they were referring to the natural language problem descriptions found in the memo fields. In the test rule file which excluded memo fields, it was found that the expert generally conjunctively linked about three or four records to arrive at a specific diagnosis. A general diagnosis involving many building systems (such as walls or structure) would require a meta-level diagnostic descriptor.

Although the development of the knowledge acquisition system hinged on the completeness of problem descriptions in each record, both the symptom and cause fields in the database were often missing. During the development phase, the necessity of record independence and completeness was stressed. Because of the lack of completeness of information in each record, patterns derived from the records proved insufficient to generate meaningful rules. Six problems with the database were identified:

- (1) The records comprised alphanumeric codes, which had to be translated into meaningful sentences each time the records were assessed.
- (2) Records were incomplete; data were insufficient to arrive at a cause for the symptom recorded on each record.
- (3) The natural language descriptors were found to be the key information sources for experts performing diagnoses; these memo fields had no syntactic formality (except that naturally ascribed to the French language).
- (4) Without a memo field, a relevant cause could be identified only if many records were linked (conjunctively).
- (5) The fields were insufficient to describe some problems (e.g. window problem) and too detailed for others (building cracks); the "grain size" was invariable yet the problem descriptions required variability.
- (6) There was no mechanism for coding spatial information.

The shortcomings of the data structure have been referred to above, but without such a knowledge acquisition and verification tool, they could not have been made explicit. A re-evaluation of the information flow during building diagnosis has identified further areas of research:

- (1) It is probable that during the translation and transformation of descriptive information too much of the initial qualitative evaluations is lost. It may be that computer representations (being unable to represent the breadth of information being processed) may be inadequate for the task,
- (2) video-based information recording, processing and retrieval systems may be required to represent information in the domain.

5.2. LEARNING ALGORITHMS

Experimentation with these learning programs has led to the inclusion of one of them in the system developed for the architectural firm. The ID3 variant was chosen for inclusion in EVITA. The structure of the database information was particularly well suited to the representation used by ID3. Attributes and attribute values could be generated from the database file headers. ID3 was also considerably faster than

AQ11. Although overnight computer runs are acceptable for academic environments, the constraints of a small to medium size architectural firm do not allow the luxury of parametric analysis with slow computer programs. As well, the ID3 algorithm is conceptually easier than AQ11 to understand and modify.

The information generated by the machine induction algorithms did not produce the anticipated results i.e. general rules about building diagnostics. There were several reasons for this. During the problem definition phase, the database size was overstated. It was originally anticipated that files containing several hundred records would be used. But the size of the architectural diagnostic files did not allow ID3 or AQ11 to discover any rules of consequence. In fact, the rules generated were extremely focussed and applied only to the particular data set from which they were generated. The data appeared to be too fine grained to adequately represent the problems occurring in the building. This situation could be improved by recognizing the linkages between various records in that database file and allowing attributes to have multiple values.

In some cases, however, over-generalization was a problem. This occurred when information, such as the concrete example mentioned above, is intended to be background knowledge but is included in the data or even absent. In such cases over-generalized rules are produced. For example, the rule

IF system = exterior wall AND orientation = 360
THEN cause = spalling

was generated. If the material happens to be concrete, then the rule holds; but if it is wood then the rule is nonsensical. To present this background knowledge in the data demands more examples or a counter-example. During analysis, the induction algorithm generated one finding not considered by experts previously. It found a pattern relating cracks in masonry walls to orientation and solar exposure.

5.3. VERIFICATION

The verification algorithms address a subset of the problems encountered when manual *ad hoc* rule generation is allowed. Only redundancies, conflicts and incompleteness arising between pairs of rules are considered here; except when searching for circular inference chains, CIRCE does not look at more complex relationships. It is therefore possible that some inconsistencies and redundancies arising between larger combinations of rules may be overlooked (Ginsberg, 1987). It is believed, however, that the very significant increase in processing time and computational resources required to perform a complete investigation make such a system inappropriate for this application, more especially since the rule-base although potentially extensive is at this stage shallow. When the knowledge-base is more fully developed, multi-level verification procedures and an optimisation capability should be implemented.

CIRCE brings logical inconsistencies to the user's attention, but does not correct them since it does not have access to the user's intentions. Determination of the logical relationships between rules is a question of syntax, but semantics may influence the appropriate modifications; e.g. where two rules reach different conclusions about a single parameter, based on the same evidence, the rules may be complementary or in conflict, depending on the context. Automatic elimination of

irrelevancies and redundancies would be possible; this may be counter-productive in some instances, e.g. where an incorrect attribute name is used. It is believed that simple notification is the safest option.

Without semantic information, the truth of a rule cannot be automatically determined; it is quite possible for a system to be logically impeccable while not modelling a process in any useful way. Verification does not constitute proof that a particular rule-based system is the best or even an acceptable solution. It is a necessary (though not sufficient) step in the process of developing a working knowledge-based system.

6. Prognosis

If a computer-based approach is to be pursued for the diagnosis of building problems, this study has identified:

- a need for a universal taxonomy and pathology for buildings; experts in the building industry must come to agreement about the syntax of the problems they describe before a production system can be built to address the problem;
- a need to better understand what spatial information is required to describe the pathology;
- a need to define a verification system that can provide partial semantic checking as well as the syntactic checking of CIRCE.

Before knowledge acquisition can be properly undertaken in the development of a system the first phase, knowledge representation, must be complete. In order to reach an informed decision about the most appropriate representation, it is essential to develop familiarity with the terms and concepts used in the domain. In practical terms this means that some initial knowledge acquisition work, *ad hoc* and informally structured, should precede detailed consideration of the representation.

The representation of diagnostic knowledge in this domain is practicable using production rules, but it requires underlying system models for acceptance. A backward chaining mechanism would be the natural inference mechanism to use during analysis of problems, but the data structure imposed, by the client in this case, implies a forward inference mechanism. The choice of inference mechanism was of less concern than the development of a system for authoring and maintaining knowledge derived from a database that did not capture the required information.

Although no rules of any consequence were generated for the architectural firm, the application of these techniques provided useful insight into the representation of building problems. It particularly identified the need for multiple record linking, meta-level diagnostic knowledge and geometric representations. By analysing the data the architects were able to determine, for the first time, the usefulness of what they had been collecting and to map portions of their procedure that worked and techniques that did not. For example, the memo fields were extremely valuable, but the zone description hierarchy was poorly conceived to represent spatial information.

The long-term use of this rule-authoring method will produce an orderly and consistent knowledge-base. The module and an associated inference engine (or knowledge-based system shell) fit the firm's existing method diagnosis, i.e. the

procedure produces an explicit record of problems. Developing new representations that better describe the building faults has begun:

- Geometric information is being coded on a generic building type to provide the expert with a model for the spatial continuity of symptoms and connection of building systems.
- Representation of failure-prone building components by fine-grained objects may provide the required variation in information grain size.

The issue of taxonomy and pathology must be addressed by the building industry as a whole. Eventually it is hoped this will lead to complete pathologies and representations supporting not only diagnosis but also design.

We gratefully acknowledge the contributions of Patrice Lapointe and his firm COPLANAM Ltee to this work. Without their expertise, this work would not have been possible.

References

- BAYAZIT, N. (1984). A study on the diagnosis of building use failure. In *Design Methods and Theories*, 2(19), 268–280.
- BOOSE, J. H. (1986). *Expertise Transfer for Expert System Design*, vol. 3. New York: Elsevier.
- DAVIDSON, C. H., DAVIDSON, P. L. & RUBERG, K. (1988). Expert systems and the use of information in building design and construction, *The Journal of Documentation*, 2(44), 91–118.
- GINSBERG, A. (1987). A new approach to checking knowledge-bases for inconsistency and redundancy. In *Proceedings of the Third Annual Expert Systems in Government Conference*, Washington D.C.
- HARRISON, S. (1987). *The Office Design Project*, Video, Reel 1 of 1, Palo Alto, CA: Xerox Parc Corp.
- HOFFMAN, R. R. (1987). The problem of extracting the knowledge of experts form the perspective of experimental psychology, *AI Magazine*, 2(8), 65–67, American Association of Artificial Intelligence, Live Oak Press.
- KAHN, G. S. KEPNER, A. & PEPPER, J. (1987). TEST: A model-driven application shell, *Proceedings of the AAAI-87, Sixth National Conference on Artificial Intelligence*, pp. 814–818. Seattle, WA.
- KELLY, G. A. (1955). *The Psychology of Personal Constructs*, New York: Norton.
- MICHALSKI, R. S. (1983). A theory and methodology of inductive learning, *Machine Learning: An Artificial Intelligence Approach*, R. S. MICHALSKI, J. G. CARBONELL & T. M. MITCHELL, Eds. pp. 83–134. Palo Alto, California: Tioga Publishers.
- MICHALSKI, R. S., MOZETIC, I., HONG, J. & LAVRAC, N. (1986). The multi-purpose incremental learning system AQ15 and its testing applications to three medical domains, *Proceedings AAAI-86, Fifth National Conference on Artificial Intelligence, Philadelphia, PA*, pp. 1041–1044, California: Morgan Kaufman Publishers.
- NGUYEN, T. A., PERKINS, W. A., LAFFEY, T. J. & PECORA, D. (1985). Checking an expert system's knowledge-base for consistency and completeness. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pp. 374–378. Menlo Park, California: American Association for Artificial Intelligence.
- PAZZANI, M. J. (1986). Refining the knowledge-base of a diagnostic expert system: an application of failure-driven learning, *Proceedings AAAI-86, Fifth National Conference on Artificial Intelligence, Philadelphia, PA*, pp. 1029–1033. California: Morgan Kaufman.

- QUINLAN, J. R. (1986). Induction of decision trees, *Machine Learning*, 1(1), 81–106. Boston, Mass: Kluwer Academic Publishers.
- SHORTLIFFE, E. H. (1976). *Computer-Based Medical Consultations: MYCIN*, New York: Elsevier.
- STACHOWITZ, R. A., COMBS, J. B. & CHANG, C. L. (1987). Validation of knowledge-based systems, *Proceedings Second AIAA/NASA/USAF Symposium on Automation, Robotics and Advanced Computing for the National Space Program*, Arlington, VA.
- SUWA, M., SCOTT, A. C. & SHORTLIFFE, E. H. (1982). An approach to verifying completeness and consistency in a rule-based expert system. *AI Magazine* 3(4), 16–21.
- TANAKA, T. (1986). Parsing circuit topology in a deductive system, *Proceedings AAAI-86, Fifth National Conference on Artificial Intelligence, Philadelphia, PA.*, pp. 1029–1033 CA: Morgan Kaufman.
- WATERMAN, D. A. (1986). *A Guide to Expert Systems*, pp. 272–289. Reading MA: Addison Wesley.
- WELBANK, M. (1983). *A Review of Knowledge Acquisition Techniques for Expert Systems*, pp. 16–49. Ipswich, England: Martlesham Consultancy Services, British Telecom Research Laboratories, Martlesham Heath.
- WILSON, F. (1984). *Building Materials Evaluation Handbook*, p. 358. New York: Van Nostrand Reinhold.
- WINSTON, P. H. & HORN, B. K. P. (1984). *Lisp*. Reading, Mass: Addison-Wesley.
- XIANG, Z., SRIHARI, S. N., SHAPIRO, S. C. & CHUTKOW, (1984). Analogical and propositional representations of structure in neurological diagnosis, *Proceedings; First Conference on Artificial Intelligence Applications*, IEEE Computer Society.