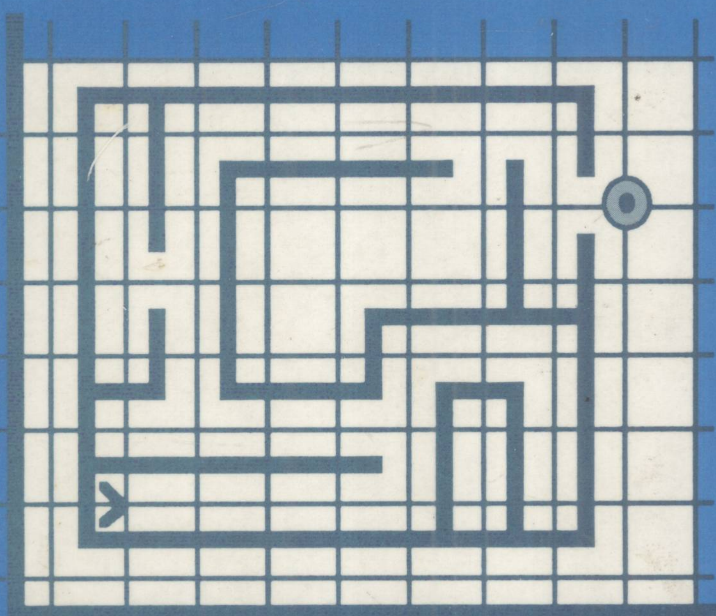
ARTIFICIAL INTELLIGENCE IN ENGINEERING: DIAGNOSIS AND LEARNING

EDITOR: J.S. GERO



ELSEVIER
COMPUTATIONAL MECHANICS PUBLICATIONS

ARTIFICIAL INTELLIGENCE IN ENGINEERING: DIAGNOSIS AND LEARNING

Edited by

J.S. Gero
University of Sydney, Australia

ELSEVIER
Amsterdam - Oxford - New York - Tokyo 1988

Co-published with

COMPUTATIONAL MECHANICS PUBLICATIONS
Southampton - Boston 1988

J.S. GERO
Department of Architectural Science
University of Sydney
NSW 2006
Australia

Associate Editor: R.A. Adey
Computational Mechanics Publications
Ashurst Lodge
Ashurst
Southampton
SO4 2AA, U.K.

Distribution of this book is being handled by:

ELSEVIER SCIENCE PUBLISHERS B.V.
Sara Burgerhartstraat 25, P.O. Box 211
1000 AE Amsterdam, The Netherlands

Distributors for the United States and Canada:

ELSEVIER SCIENCE PUBLISHING COMPANY INC.
52 Vanderbilt Avenue
New York, N.Y. 10017, U.S.A.

British Library Cataloguing in Publication Data
Artificial intelligence in engineering:
diagnosis and learning.
1. Engineering. Applications of
artificial intelligence
I. Gero, John S. (John Steven)
620'0078'563
ISBN 1-85312-012-X

Library of Congress Catalog Card number 88-71238

ISBN 0-444-70471-X Elsevier Science Publishers B.V.
ISBN 0-444-70466-3(Set)
ISBN 1-85312-012-X Computational Mechanics Publications UK
ISBN 0-931215-99-4 Computational Mechanics Publications USA

Published by:

COMPUTATIONAL MECHANICS PUBLICATIONS
Ashurst Lodge, Ashurst
Southampton , SO4 2AA, U.K.

Printed in Great Britain by The Bath Press Limited, Avon

NON SOLUS
ELSEVIER

COMPUTATIONAL
MECHANICS
PUBLICATIONS

Acknowledgement is made to R. Lelouche and P. Dion for the use of Figure 6 on page 414, which appears on the front cover of this book.

PREFACE

The International Conference on Applications of Artificial Intelligence in Engineering series is establishing itself as the unique forum for the presentation of the latest research, development and application of artificial intelligence in engineering. It covers all the engineering fields. The first conference was held in Southampton, UK in 1986. The second conference was held in Cambridge, Massachusetts, USA in 1987. The third conference was held in Palo Alto, California, USA in August 1988. This volume and its two companion volumes contain the papers presented at the third conference.

Computers and computing have become an integral part of engineering. Computers are used extensively to carry out analyses and occasionally in low level design such as member selection in structural engineering. They are being increasingly used to produce drawings and other graphical representations to communicate ideas. Most engineering offices also use computers for office management, job control and word processing. All of these applications are built on two fundamental concepts. The first concept is that the world of interest can be described by the "calculus of real numbers—algebra". The second concept has to do with computing itself and is concerned with the way we make computers work, through instructions to the computer codified as "procedures" which describe what must be done. These we call computer programs—a set of detailed instructions which are executed sequentially unless the program contains instructions which alter this sequential flow. Whilst this is a concept deeply embedded in traditional computing it has important ramifications for users. Such computing is called von Neumann computing after the inventor of this architecture for computing.

Engineering is concerned with much more than calculation based on mathematical descriptions of the world. Professional engineering is concerned with concepts, ideas, judgment and experience. All of these appear to be outside the realm of traditional computing. Human beings discourse with each other using models of the worlds largely unrelated to either mathematical descriptions or procedural representations. They make use of knowledge about objects, events and processes and make declarative statements about them. These are often written down symbolically. The limits of traditional computing are that it is unable to represent and manipulate knowledge in an explicit and coherent form and that it is unable readily to perform symbolic computation.

Artificial intelligence is largely concerned with the acquisition, representation and manipulation of human knowledge in symbolic form. Human knowledge is thought of as being reasoning (rather than the simple ability to acquire facts as you might find in an encyclopedia). Just as the industrial revolution can be considered to have automated mechanical power, and the computer revolution to have automated calculation, so artificial intelligence automates symbolic reasoning.

Barr and Feigenbaum in *The Handbook of Artificial Intelligence* could well have been writing about professional engineering knowledge when they stated: "Since there are no mathematical cores to structure the calculational use of the computer, such areas will inevitably be served by symbolic models and symbolic inference (reasoning) techniques."

Diagnosis and learning are two of the important areas of artificial intelligence of interest to engineering. Diagnosis derives its name from the early expert systems work related to diagnosing medical disorders. In engineering it has come to mean both diagnosis (in the medical sense) and evaluation or analysis (in the engineering sense). The use of symbolic reasoning has substantially expanded the functions and roles of diagnosis in engineering. This is exemplified by this collection of papers. Learning is a novel area of computer application in engineering and has no counterpart in traditional engineering but is becoming an important area.

The papers in this volume are presented under the following headings.

Diagnosis from Structure and Behaviour
Integrated Diagnostic Reasoning
Diagnosis as Control
Diagnosis Processes and Environments
Learning and Tutoring

A large number of papers were submitted for consideration. Each paper was refereed by three referees. This task was ably executed with the aid of the International Advisory Board listed below. Members of the board worked assiduously to select appropriate papers, thanks are due to them.

The final manuscript was sub-edited by Fay Sudweeks to produce a degree of uniformity lacking in the submissions.

John S. Gero
University of Sydney

CONTENTS

SECTION ONE

Diagnosis from Structure and Behaviour

Extensions to ATMS-based diagnosis

P. Struss
Siemens Corp., Munich, West Germany

ABSTRACT In particular in technical domains, knowledge-based diagnosis systems require an explicit representation of the structure and behavior of the device or system under consideration. A system was developed which allows both a hierarchical and a view-oriented structuring of component models. Thus, it provides a basis for a selective instantiation and execution of these models in order to express a focus of attention and reduce complexity. Diagnosis is regarded as an inference process in which contradictions between assumptions about the correct or intended behavior of a system and the observed behavior are detected and exploited in order to identify the wrong assumptions. Hence, the diagnosis system uses an Assumption-Based Truth-Maintenance System (ATMS) which additionally allows to handle multiple test situations and assumptions about the correctness of observations and measurements.

1 INTRODUCTION

In technical devices, interrelations of components and parameters appear to be exact and certain. Physical laws need no guesses. However, tasks of reasoning about them, such as diagnosis, include lots of inferences which are grounded on hypotheses and retractable assumptions.

For diagnosis, the discrepancy between expectations and observations of the behavior of a device is essential. Detecting the **necessity** of a diagnosis task requires only a naive expectation of a specific function of the device (say, a tv). When performing it, however, which means identifying the malfunctioning part(s), the expectations about the behavior have to be based on a model of the device and involve the **assumption** that all the parts work properly. On the other hand, diagnostic reasoning includes investigations whether an **assumed** misbehavior of a specific part could explain the observed symptoms. Moreover, **assumptions** about the reliability of observations and measurements are involved in diagnosis, and so are **hypotheses** about likely faults etc.

Different sets of such assumptions establish different consistent "possible worlds" which potentially contradict each other. For example, the propaganda brochure about our tv and the rustling thing with the foggy screen in front of us are such conflicting "possible worlds". Hence, exploring (maximal) consistent contexts and detecting (minimal) sets of conflicting assumptions appears to be a useful task in a diagnostic problem solving systems. A tool for doing this is the Assumption-based Truth Maintenance System, **ATMS** (deKleer [3], deKleer [4]). DeKleer-Williams [5] proposes a general framework for model-based diagnosis which heavily exploits the ATMS. This approach is sketched in section 2.

Several problems, however, which occur when attempting to apply this approach to realistic domains are not addressed in this paper or not solved in detail. Among these are issues of

- dealing with complexity of devices by providing means for structuring the models and for focusing analysis
- treating the performance of different tests and considerations about the precision and reliability of observations and measurements
- using fault models of the involved components.

It turns out that the basic framework can be extended to include solutions to the first and the second problem that appear to be necessary conditions for approaching real applications. These extensions are described in sections 3 and 4.

2 ATMS-BASED DIAGNOSIS

2.1 The Basic ATMS

The ATMS is a tool which, independent of the application domain and of the specific form of the inference mechanism used by a problem solver, records the dependencies of its inferences including the assumptions supporting them. Hence, the (basic) ATMS is not an inference mechanism in itself, it is only supplied with the conclusions derived by the problem solver together with their direct conditions. In particular, for each problem solver datum, the ATMS constructs information about the (minimal) sets of assumptions which suffice to derive it.

In the following, some basic concepts of the ATMS are briefly introduced. For details see deKleer [3], de Kleer [4].

- An **ATMS node** is created by the ATMS for each problem solver datum passed to it. The nodes serve as the units for gathering all the dependency information the ATMS keeps track of. Regardless of how the problem solver represents and structures its data, the ATMS treats them as atomic, unstructured propositional expressions. There is a special node **False**, denoting a contradiction.

- A **justification** refers to a single inference step of the problem solver the ATMS is supplied with. It links the *supported node* (corresponding to the datum derived by the inference) with the *supporting nodes* (the data which are the direct preconditions for the inference step) and is given a unique identifier. E.g. a rule (or constraint) saying "IF x=5 and y=10 then z=15" produces the justification shown in Fig. 1.

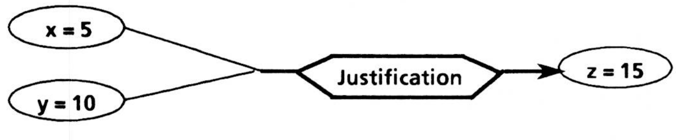


Figure 1

- An **assumption** represents the decision of the problem solver to believe a certain datum. Thus an assumption is clearly distinct from the assumed node it supports.

- An **environment** of a node is a set of assumptions from which the node can be inferred using the existing justifications. E.g. if we entered assumptions $Ass_{x=5}$, $Ass_{y=10}$ for x=5 and y=10, they would constitute an environment of z=15. The empty set is a distinguished environment which represents "**True**", because in this case a node can be derived without making any assumptions. A **minimal environment** by definition does not contain any other environment as a proper subset. **Nogoods** are inconsistent environments, i.e. environments of the node *False*.

- The **label** of a node is the set of its minimal environments, with the nogoods removed, and thus represents a disjunction of the consistent sets of ultimate preconditions of the node. If the label is the empty set, then the existing assumptions and justifications do not allow the node to be derived. A label containing the empty environment (necessarily only this one) indicates a **fact**. The notion of a label can be extended to labels of justifications in an obvious way.

After each inference step of the problem solver which is transferred to the ATMS, the label of each node describes the minimal conditions for the corresponding datum in terms of sets of assumptions. Data which are contradictory, i.e. which together imply *False*, are characterized by labels which cannot be consistently combined, as are the derivations dependent on them. Thus, the problem solver is able to investigate different contexts, established by different sets of assumptions, in parallel without mixing them up and without having to re-derive conclusions which are valid in their intersections.

2.2 ATMS-Based Diagnosis

A possible way of diagnosing systems with a known structure established by combined components with a predictable behavior is the following: assume a correct behavior and then exploit the discrepancies between the consequences of these assumptions and the real, observed behavior in order to falsify correctness assumptions w.r.t. one or more components. Obviously, a diagnosis system following this line needs some explicit representation of structure and behavior, a deep model of the device under consideration. Furthermore, because of its capabilities of keeping track of assumption-based inferences and inconsistencies, the use of an ATMS appears to be of help for the task.

DeKleer-Williams [5] proposes a general diagnosis system which can be summarized as follows:

The components' behavior is modelled by constraints between input and output variables. For each component, an assumption is created stating that it is working properly. If information about values of variables is provided by measurements and passed to the system as facts, the constraints of the network will infer values for other variables. The results of these inferences are transferred to the ATMS in terms of justifications whose list of antecedents does not only contain the values which were used in the computation but also the correctness assumption of the respective component.

Due to the mechanisms provided by the ATMS, the label of a value contains sets of those correctness assumptions which were involved in its derivation. Whenever a contradiction occurs, i.e. the system computes conflicting values for a variable, **nogoods** are established which in this case are minimal sets of correctness assumptions which cannot all be true. These so-called **conflict sets** are used to construct **candidate sets**, i.e. minimal sets of suspected components whose malfunctioning could account for all detected conflicts.

Using the standard example of a circuit containing adders (A_i) and multipliers (M_i) (Fig. 2), the mechanism can be demonstrated: Supplied with the input A=3, B=2, C=2, D=3 and E=3, the constraint network produces the expected output F=12 and G=12.