

# Encyclopedia of Artificial Intelligence

Juan Ramón Rabuñal Dopico  
*University of A Coruña, Spain*

Julián Dorado de la Calle  
*University of A Coruña, Spain*

Alejandro Pazos Sierra  
*University of A Coruña, Spain*

Information Science  
**REFERENCE**

**INFORMATION SCI**

Hershey • New York

Director of Editorial Content: Kristin Klinger  
Managing Development Editor: Kristin Roth  
Development Editorial Assistant: Julia Mosemann, Rebecca Beistline  
Senior Managing Editor: Jennifer Neidig  
Managing Editor: Jamie Snavelly  
Assistant Managing Editor: Carole Coulson  
Typesetter: Jennifer Neidig, Amanda Appicello, Cindy Consonery  
Cover Design: Lisa Tosheff  
Printed at: Yurchak Printing Inc.

Published in the United States of America by  
Information Science Reference (an imprint of IGI Global)  
701 E. Chocolate Avenue, Suite 200  
Hershey PA 17033  
Tel: 717-533-8845  
Fax: 717-533-8661  
E-mail: [cust@igi-global.com](mailto:cust@igi-global.com)  
Web site: <http://www.igi-global.com/reference>

and in the United Kingdom by  
Information Science Reference (an imprint of IGI Global)  
3 Henrietta Street  
Covent Garden  
London WC2E 8LU  
Tel: 44 20 7240 0856  
Fax: 44 20 7379 0609  
Web site: <http://www.eurospanbookstore.com>

Copyright © 2009 by IGI Global. All rights reserved. No part of this publication may be reproduced, stored or distributed in any form or by any means, electronic or mechanical, including photocopying, without written permission from the publisher.

Product or company names used in this set are for identification purposes only. Inclusion of the names of the products or companies does not indicate a claim of ownership by IGI Global of the trademark or registered trademark.

#### Library of Congress Cataloging-in-Publication Data

Encyclopedia of artificial intelligence / Juan Ramon Rabunal Dopico, Julian Dorado de la Calle, and Alejandro Pazos Sierra, editors.  
p. cm.

Includes bibliographical references and index.

Summary: "This book is a comprehensive and in-depth reference to the most recent developments in the field covering theoretical developments, techniques, technologies, among others"--Provided by publisher.

ISBN 978-1-59904-849-9 (hardcover) -- ISBN 978-1-59904-850-5 (ebook)

I. Artificial intelligence--Encyclopedias. I. Rabunal, Juan Ramon, 1973- II. Dorado, Julian, 1970- III. Pazos Sierra, Alejandro.

Q334.2.E63 2008

006.303--dc22

2008027245

#### British Cataloguing in Publication Data

A Cataloguing in Publication record for this book is available from the British Library.

All work contributed to this encyclopedia set is new, previously-unpublished material. The views expressed in this encyclopedia set are those of the authors, but not necessarily of the publisher.

*If a library purchased a print copy of this publication, please go to <http://www.igi-global.com/agreement> for information on activating the library's complimentary electronic access to this publication.*

# Knowledge-Based Systems

**Adrian A. Hopgood**

*De Montfort University, UK*

**K**

## INTRODUCTION

The tools of artificial intelligence (AI) can be divided into two broad types: knowledge-based systems (KBSs) and computational intelligence (CI). KBSs use explicit representations of knowledge in the form of words and symbols. This explicit representation makes the knowledge more easily read and understood by a human than the numerically derived implicit models in computational intelligence.

KBSs include techniques such as rule-based, model-based, and case-based reasoning. They were among the first forms of investigation into AI and remain a major theme. Early research focused on specialist applications in areas such as chemistry, medicine, and computer hardware. These early successes generated great optimism in AI, but more broad-based representations of human intelligence have remained difficult to achieve (Hopgood, 2003; Hopgood, 2005).

## BACKGROUND

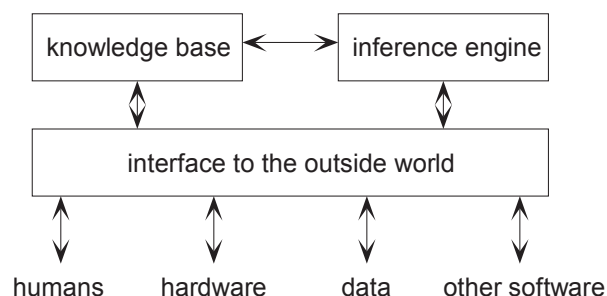
The principal difference between a knowledge-based system and a conventional program lies in its structure. In a conventional program, domain knowledge is intimately intertwined with software for controlling the

application of that knowledge. In a knowledge-based system, the two roles are explicitly separated. In the simplest case there are two modules: the knowledge base and the control module is called the inference engine. Some interface capabilities are also required for a practical system, as shown in Figure 1.

Within the knowledge base, the programmer expresses information about the problem to be solved. Often this information is declarative, i.e. the programmer states some facts, rules, or relationships without having to be concerned with the detail of how and when that information should be applied. These latter details are determined by the inference engine, which uses the knowledge base as a conventional program uses a data file. A KBS is analogous to the human brain, whose control processes are approximately unchanging in their nature, like the inference engine, even though individual behavior is continually modified by new knowledge and experience, like updating the knowledge base.

As the knowledge is represented explicitly in the knowledge base, rather than implicitly within the structure of a program, it can be entered and updated with relative ease by domain experts who may not have any programming expertise. A knowledge engineer is someone who provides a bridge between the domain

*Figure 1. The main components of a knowledge-based system*



expertise and the computer implementation. The knowledge engineer may make use of meta-knowledge, i.e. knowledge about knowledge, to ensure an efficient implementation.

Traditional knowledge engineering is based on models of human concepts. However, it has recently been argued that animals and pre-linguistic children operate effectively in a complex world without necessarily using concepts. Moss (2007) has demonstrated that agents using non-conceptual reasoning can outperform stimulus-response agents in a grid-world test bed. These results may justify the building of non-conceptual models before moving on to conceptual ones.

## TYPES OF KNOWLEDGE-BASED SYSTEM

### Expert Systems

Expert systems are a type of knowledge-based system designed to embody expertise in a particular specialized domain such as diagnosing faulty equipment (Yanga, 2005). An expert system is intended to act like a human expert who can be consulted on a range of problems within his or her domain of expertise. Typically, the user of an expert system will enter into a dialogue in which he or she describes the problem – such as the symptoms of a fault – and the expert system offers advice, suggestions, or recommendations. It is often proposed that an expert system must offer certain capabilities that mirror those of a human consultant. In particular, it is often stated that an expert system must be capable of justifying its current line of inquiry and explaining its reasoning in arriving at a conclusion. This functionality can be integrated into the inference engine (Figure 1).

### Rule-Based Systems

Rules are one of the most straightforward means of representing knowledge in a KBS. The simplest type of rule is called a production rule and takes the form:

if <condition> then <conclusion>

An example production rule concerning a boiler system might be:

```
/* rule1 */
if valve is open and flow is high then steam is escaping
```

Part of the attraction of using production rules is that they can often be written in a form that closely resembles natural language, as opposed to a computer language. The facts in a KBS for boiler monitoring might include:

```
/* fact1 */
valve is open
```

```
/* fact2 */
flow is high
```

One or more given facts may satisfy the condition of a rule, resulting in the generation of a new fact, known as a derived fact. For example, by applying rule1 to fact1 and fact2, fact3 can be derived:

```
/* fact3 */
steam is escaping
```

The derived fact may satisfy the condition of another rule, such as:

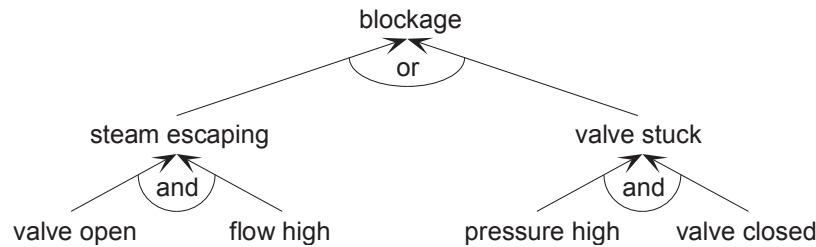
```
/* rule2 */
if steam is escaping or valve is stuck then outlet is blocked
```

This, in turn, may lead to the generation of a new derived fact or an action. Rule1 and rule2 are inter-dependent, since the conclusion of one can satisfy the condition of the other. The inter-dependencies amongst the rules define a network, as shown in Figure 2, known as an inference network.

It is the job of the inference engine to traverse the inference network to reach a conclusion. Two important types of inference engine can be distinguished: forward-chaining and backward-chaining, also known as data-driven and goal-driven, respectively. A KBS working in data-driven mode takes the available information, i.e. the given facts, and generates as many derived facts as it can. In goal-driven mode, evidence is sought to support a particular goal or proposition.

The data-driven (forward chaining) approach might typically be used for problems of interpretation, where the aim is to find out whatever the system can infer about some data. The goal-driven (backward chaining)

Figure 2. An inference network for a boiler system



approach is appropriate when a more tightly focused solution is required, such as the generation of a plan for a particular goal. In the example of a boiler monitoring system, forward chaining would lead to the reporting of any recognised problems. In contrast, backward chaining might be used to diagnose a specific mode of failure by linking a logical sequence of inferences, disregarding unrelated observations.

The rules that make up the inference network in Figure 2 are used to link cause and effect:

if <cause> then <effect>

Using the inference network, an inference can be drawn that if the valve is open and the flow rate is high (the causes) then steam is escaping (the effect). This is the process of *deduction*. Many problems, such as diagnosis, involve reasoning in the reverse direction, i.e. the user wants to ascertain a cause, given an effect. This is *abduction*. Given the observation that steam is escaping, abduction can be used to infer that valve is open and the flow rate is high. However, this is only a valid conclusion if the inference network shows *all* of the circumstances in which steam may escape. This is the closed-world assumption.

If many examples of cause and effect are available, the rule (or inference network) that links them can be inferred. For instance, if every boiler blockage ever seen was accompanied by steam escaping and a stuck valve, then rule2 above might be inferred from those examples. Inferring a rule from a set of example cases of cause and effect is termed *induction*.

Hopgood (2001) summarizes deduction, abduction, and induction as follows:

- deduction: cause + rule  $\Rightarrow$  effect
- abduction: effect + rule  $\Rightarrow$  cause
- induction: cause + effect  $\Rightarrow$  rule

## Logic Programming

Logic programming describes the use of logic to establish the truth, or otherwise, of a proposition. It is, therefore, an underlying principle for rule-based systems. Although various forms of logic programming have been explored, the most commonly used one is the Prolog language (Bramer, 2005), which embodies the features of backward chaining, pattern matching, and list manipulation.

The Prolog language can be programmed declaratively, although an appreciation of the procedural behavior of the language is needed in order to program it effectively. Prolog is suited to symbolic problems, particularly logical problems involving relationships between items. It is also suitable for tasks that involve data lookup and retrieval, as pattern-matching is fundamental to the functionality of the language.

## Symbolic Computation

A knowledge base may contain a mixture of numbers, letters, words, punctuation, and complete sentences. These symbols need to be recognised and processed by the inference engine. Lists are a particularly useful

data structure for symbolic computation, and they are integral to the AI languages Lisp and Prolog. Lists allow words, numbers, and symbols to be combined in a wide variety of ways. A list in the Prolog language might look like this:

```
[animal, [cat, dog], vegetable, mineral]
```

where this example includes a nested list, i.e. a list within a list. In order to process lists or similar structures, the technique of pattern matching is used. For example, the above list in Prolog could match to the list

```
[animal, [_ , X], vegetable, Y]
```

where the variables X and Y would be assigned values of `dog` and `mineral` respectively. This pattern matching capability is the basis of an inference engine's ability to process rules, facts and evolving knowledge.

## Uncertainty

The examples considered so far have all dealt with unambiguous facts and rules, leading to clear conclusions. In real life, the situation can be complicated by three forms of uncertainty:

### Uncertainty in the Rule Itself

For example, rule 1 (above) stated that an open valve and high flow rate lead to an escape of steam. However, if the boiler has entered an unforeseen mode, it made be that these conditions do not lead to an escape of steam. The rule ought really to state that an open valve and high flow rate will *probably* lead to an escape of steam.

### Uncertainty in the Evidence

There are two possible reasons why the evidence upon which the rule is based may be uncertain. First, the evidence may come from a source that is not totally reliable. For example, in rule 1 there may be an element of doubt whether the flow rate is high, as this information relies upon a meter of unspecified reliability. Second, the evidence itself may have been derived by a rule whose conclusion was probable rather than certain.

## Use of Vague Language

Rule 1, above, is based around the notion of a "high" flow rate. There is uncertainty over whether "high" means a flow rate of the order of  $1\text{cm}^3\text{s}^{-1}$  or  $1\text{m}^3\text{s}^{-1}$ .

Two popular techniques for handling the first two sources of uncertainty are Bayesian updating and certainty theory (Hopgood, 2001). Bayesian updating has a rigorous derivation based upon probability theory, but its underlying assumptions, e.g., the statistical independence of multiple pieces of evidence, may not be true in practical situations. Certainty theory does not have a rigorous mathematical basis, but has been devised as a practical and pragmatic way of overcoming some of the limitations of Bayesian updating. It was first used in the classic MYCIN system for diagnosing infectious diseases (Buchanan, 1984). Other approaches are reviewed in (Hopgood, 2001), where it is also proposed that a practical non-mathematical approach is to treat rule conclusions as hypotheses that can be confirmed or refuted by the actions of other rules. Possibility theory, or fuzzy logic, allows the third form of uncertainty, i.e. vague language, to be used in a precise manner.

## Decision Support and Analysis

Decision support and analysis (DSA) and decision support systems (DSSs) describe a broad category of systems that involve generating alternatives and selecting among them. Web-based DSA, which uses external information sources, is becoming increasingly important. Decision support systems that use artificial intelligence techniques are sometimes referred to as intelligent DSSs.

One clearly identifiable family of intelligent DSS is expert systems, described above. An expert system may contain a mixture of simple rules based on experience and observation, known as heuristic or shallow rules, and more fundamental or deep rules. For example, an expert system for diagnosing car breakdowns may contain a heuristic that suggests checking the battery if the car will not start. In contrast, the expert system might also contain deep rules, such as Kirchoff's laws, which apply to any electrical circuit and could be used in association with other rules and observations to diagnose any electrical circuit. Heuristics can often

provide a useful shortcut to a solution, but lack the adaptability of deep knowledge.

Building and maintaining a reliable set of cause–effect pairs in the form of rules can be a huge task. The principle of model-based reasoning (MBR) is that, rather than storing a huge collection of symptom–cause pairs in the form of rules, these pairs can be *generated* by applying underlying principles to the model. The model may describe any kind of system, including systems that are physical (Fenton, 2001), software-based (Mateis, 2000), medical (Montani, 2003), legal (Bruninghaus, 2003), and behavioral (De Koning, 2000). Models of physical systems are made up of fundamental components such as tubes, wires, batteries, and valves. As each of these components performs a fairly simple role, it also has a simple failure mode. Given a model of how these components operate and interact to form a device, faults can be diagnosed by determining the effects of local malfunctions on the overall device.

Case-based reasoning (CBR) also has a major role in DSA. A characteristic of human intelligence is the ability to recall previous experience whenever a similar problem arises. This is the essence of case-based reasoning (CBR), in which new problems are solved by adapting previous solutions to old problems (Bergmann, 2003).

Consider the example of diagnosing a broken-down car. If an expert system has made a successful diagnosis of the breakdown, given a set of symptoms, it can file away this information for future use. If the expert system is subsequently presented with details of another broken-down car of exactly the same type, displaying exactly the same symptoms in exactly the same circumstances, then the diagnosis can be completed simply by recalling the previous solution. However, a full description of the symptoms and the environment would need to be very detailed, and it is unlikely to be reproduced exactly. What is needed is the ability to identify a previous case, the solution of which can be reused or modified to reflect the slightly altered circumstances, and then saved for future use. Such an approach is a good model of human reasoning. Indeed case-based reasoning is often used in a semi-automated manner, where a human can intervene at any stage in the cycle.

## FUTURE TRENDS

While large corporate knowledge-based systems remain important, small embedded intelligent systems have also started to appear in the home and workplace. Examples include washing machines that incorporate knowledge-based control and wizards for personal computer management. By being embedded in their environment, such systems are less reliant on human data input than traditional expert systems, and often make decisions entirely based on sensor data.

If AI is to become more widely situated into everyday environments, it needs to become smaller, cheaper, and more reliable. The next key stage in the development of AI is likely to be a move towards *embedded* AI, i.e. intelligent systems that are embedded in machines, devices, and appliances. The work of Choy (2003) is significant in this respect, as it demonstrates that the DARBS blackboard system can be ported to a compact platform of parallel low-cost processors.

In addition to being distributed in their applications, intelligent systems are also becoming distributed in their method of implementation. Complex problems can be divided into subtasks that can be allocated to specialized collaborative agents, bringing together the best features of knowledge-based and computation intelligence approaches (Li, 2003). As the collaborating agents need not necessarily reside on the same computer, an intelligent system can be both distributed and hybridized (Choy, 2004). Paradoxically, there is also a sense in which intelligent systems are becoming more integrated, as software agents share access to a single definitive copy of data or knowledge, accessible via the web.

## CONCLUSION

As with any technique, knowledge-based systems are not suitable for all types of problems. Each problem calls for the most appropriate tool, but knowledge-based systems can be used for many problems that would be impracticable by other means. They have been particularly successful in narrow specialist domains. Building an intelligent system that can make sensible decisions about unfamiliar situations in everyday, non-specialist domains remains a severe challenge.

This development will require progress in simulating behaviors that humans take for granted – specifically perception, recognition, language, common sense, and adaptability. To build an intelligent system that spans the breadth of human capabilities is likely to require a hybrid approach using a combination of artificial intelligence techniques.

## REFERENCES

- Bergmann, R., Althoff, K.-D., Breen, S., Göker, M., Manago, M., Traphöner, R., and Wess, S. (2003). Developing Industrial Case-Based Reasoning Applications – the INRECA Methodology (2nd Edition). Lecture Notes in Artificial Intelligence, Vol. 1612. Springer - Buchreihe.
- Bramer, M.A. (2005), Logic Programming with Prolog. Springer-Verlag, London.
- Bruninghaus, S. and Ashley, K. D. (2003). Combining case-based and model-based reasoning for predicting the outcome of legal cases. Lecture Notes in Artificial Intelligence, 2689, 65-79.
- Buchanan, B. G. and Shortliffe, E. H. (1984). Rule-Based Expert Systems: the MYCIN experiments of the Stanford Heuristic Programming Project, Addison-Wesley.
- Choy, K.W., Hopgood, A.A., Nolle, L. and O’Neill, B.C. (2003). Design and implementation of an inter-process communication model for an embedded distributed processing network. International Conference on Software Engineering Research and Practice (SERP’03), Las Vegas, 239-245.
- Choy, K.W., Hopgood, A.A., Nolle, L. and O’Neill, B.C. (2004). Implementation of a tileworld testbed on a distributed blackboard system. 18<sup>th</sup> European Simulation Multiconference (ESM2004), Magdeburg, Germany, 129-135.
- De Koning, K., Bredeweg, B., Breuker, J., and Wielinga, B. (2000). Model-based reasoning about learner behaviour. Artificial Intelligence, 117, 173-229.
- Fenton, W. G., McGinnity, T. M., and Maguire, L. P. (2001). Fault diagnosis of electronic systems using intelligent techniques: a review. IEEE Transactions on Systems Man and Cybernetics Part C - Applications and Reviews, 31, 269-281.
- Hopgood, A. A. (2001). Intelligent Systems for Engineers and Scientists, 2nd edition. CRC Press, Boca Raton.
- Hopgood, A. A. (2003). Artificial intelligence: hype or reality? IEEE Computer, 6, 24-28.
- Hopgood, A.A. (2005). The state of artificial intelligence. Advances in Computers, 65, 1-75.
- Li, G., Hopgood, A.A. and Weller, M.J. (2003). Shifting Matrix Management: a model for multi-agent cooperation. Engineering Applications of Artificial Intelligence, 16, 191-201.
- Mateis, C., Stumptner, M., and Wotawa, F. (2000). Locating bugs in Java programs - First results of the Java diagnosis experiments project. Lecture Notes in Artificial Intelligence, 1821, 174-183.
- Montani, S., Magni, P., Bellazzi, R., Larizza, C., Roudsari, A. V., and Carson, E. R. (2003). Integrating model-based decision support in a multi-modal reasoning system for managing type 1 diabetic patients. Artificial Intelligence in Medicine, 29, 131-151.
- Moss, N.G., Hopgood, A.A. and Weller, M.J. (2007). Can Agents without Concepts think? An Investigation using a Knowledge Based System. Proc. AI-2007: 27th SGAI International Conference on Artificial Intelligence, Cambridge, UK.
- Yanga, B.S., Limb, D.S., and Tanc, A.C.C. (2005). VIBEX: an expert system for vibration fault diagnosis of rotating machinery using decision tree and decision table. Expert Systems with Applications, 28(4), 735-742.

## KEY TERMS

**Backward Chaining:** Rules are applied through depth-first search of the rule base to establish a goal. If a line of reasoning fails, the inference engine must backtrack and search a new branch of the search tree. This process is repeated until the goal is established or all branches have been explored.

**Case-Based Reasoning:** Solving new problems by adapting solutions that were previously used to solve old problem.

**Closed-World Assumption:** The assumption that all knowledge about a domain is contained in the knowledge base. Anything that is not true according to the knowledge base is assumed to be false.

**Deep Knowledge:** Fundamental knowledge with general applicability, such as the laws of physics, which can be used in conjunction with other deep knowledge to link evidence and conclusions.

**Forward Chaining:** Rules are applied iteratively whenever their conditions are satisfied, subject to a selection mechanism known as conflict resolution when the conditions of multiple rules are satisfied.

**Heuristic or Shallow Knowledge:** Knowledge, usually in the form of a rule, that links evidence and conclusions in a limited domain. Heuristics are based on observation and experience, without an underlying derivation or understanding.

**Inference Network:** The linkages between a set of conditions and conclusions.

**Knowledge-Based System:** System in which the *knowledge base* is explicitly separated from the *inference engine* that applies the knowledge.

**Model-Based Reasoning:** The knowledge base comprises a model of the problem area, constructed from component parts. The inference engine reasons about the real world by exploring behaviors of the model.

**Production Rule:** A rule of the form if <condition> then <conclusion>.