WIKIPEDIA

Expert system

In artificial intelligence, an **expert system** is a computer system emulating the decision-making ability of a human expert.[1] Expert systems are designed to solve complex problems by reasoning through bodies of knowledge, represented mainly as if–then rules rather than through conventional procedural code.[2] The first expert systems were created in the 1970s and then proliferated in the 1980s.[3] Expert systems were among the first truly successful forms of artificial intelligence (AI) software.[4][5][6][7][8] An expert system is divided into two subsystems: the inference engine and the knowledge base. The knowledge base represents facts and rules. The inference engine applies the rules to the known facts to deduce new facts. Inference engines can also include explanation and debugging abilities.

A Symbolics Lisp Machine: an early platform for expert systems

Contents

History

Early development

Soon after the dawn of modern computers in the late 1940s – early 1950s, researchers started realizing the immense potential these machines had for modern society. One of the first challenges was to make such machine capable of “thinking” like humans. In particular, making these machines capable of making important decisions the way humans do. The medical / healthcare field presented

the tantalizing challenge to enable these machines to make medical diagnostic decisions.[9]

Thus, in the late 1950s, right after the information age had fully arrived, researchers started experimenting with the prospect of using computer technology to emulate human decision-making. For example, biomedical researchers started creating computer-aided systems for diagnostic applications in medicine and biology. These early diagnostic systems used patients' symptoms and laboratory test results as inputs to generate a diagnostic outcome.[10][11] These systems were often described as the early forms of expert systems. However, researchers realized that there were significant limitations when using traditional methods such as flow-charts[12] [13] statistical pattern-matching,[14] or probability theory.[15][16]

Formal introduction & later developments

This previous situation gradually led to the development of expert systems, which used knowledge-based approaches. These expert systems in medicine were the MYCIN expert system,[17] the INTERNIST-I expert system[18] and later, in the middle of the 1980s, the CADUCEUS.[19]

Expert systems were formally introduced around 1965[20] by the Stanford Heuristic Programming Project led by Edward Feigenbaum, who is sometimes termed the "father of expert systems"; other key early contributors were Bruce Buchanan and Randall Davis. The Stanford researchers tried to identify domains where expertise was highly valued and complex, such as diagnosing infectious diseases (Mycin) and identifying unknown organic molecules (Dendral). The idea that "intelligent systems derive their power from the knowledge they possess rather than from the specific formalisms and inference schemes they use"[21] – as Feigenbaum said – was at the time a significant step forward, since the past research had been focused on heuristic computational methods, culminating in attempts to develop very general-purpose problem solvers (foremostly the conjunct work of Allen Newell and Herbert Simon).[22] Expert systems became some of the first truly successful forms of artificial intelligence (AI) software.[4][5][6][7][8]

Research on expert systems was also active in France. While in the US the focus tended to be on rules-based systems, first on systems hard coded on top of LISP programming environments and then on expert system shells developed by vendors such as Intellicorp, in France research focused more on systems developed in Prolog. The advantage of expert system shells was that they were somewhat easier for nonprogrammers to use. The advantage of Prolog environments was that they were not focused only on *if-then* rules; Prolog environments provided a much better realization of a complete first order logic environment.[23][24]

In the 1980s, expert systems proliferated. Universities offered expert system courses and two-thirds of the Fortune 500 companies applied the technology in daily business activities.[3][25] Interest was international with the Fifth Generation Computer Systems project in Japan and increased research funding in Europe.

In 1981, the first IBM PC, with the PC DOS operating system, was introduced. The imbalance between the high affordability of the relatively powerful chips in the PC,

compared to the much more expensive cost of processing power in the mainframes that dominated the corporate IT world at the time, created a new type of architecture for corporate computing, termed the client–server model.[26] Calculations and reasoning could be performed at a fraction of the price of a mainframe using a PC. This model also enabled business units to bypass corporate IT departments and directly build their own applications. As a result, client-server had a tremendous impact on the expert systems market. Expert systems were already outliers in much of the business world, requiring new skills that many IT departments did not have and were not eager to develop. They were a natural fit for new PC-based shells that promised to put application development into the hands of end users and experts. Until then, the main development environment for expert systems had been high end Lisp machines from Xerox, Symbolics, and Texas Instruments. With the rise of the PC and client-server computing, vendors such as Intellicorp and Inference Corporation shifted their priorities to developing PC-based tools. Also, new vendors, often financed by venture capital (such as Aion Corporation, Neuron Data, Exsys, and many others[27][28]), started appearing regularly.

The first expert system to be used in a design capacity for a large-scale product was the SID (Synthesis of Integral Design) software program, developed in 1982. Written in LISP, SID generated 93% of the VAX 9000 CPU logic gates.[29] Input to the software was a set of rules created by several expert logic designers. SID expanded the rules and generated software logic synthesis routines many times the size of the rules themselves. Surprisingly, the combination of these rules resulted in an overall design that exceeded the capabilities of the experts themselves, and in many cases out-performed the human counterparts. While some rules contradicted others, top-level control parameters for speed and area provided the tie-breaker. The program was highly controversial but used nevertheless due to project budget constraints. It was terminated by logic designers after the VAX 9000 project completion.

During the years before the middle of the 1970s, the expectations of what expert systems can accomplish in many fields tended to be extremely optimistic. At the beginning of these early studies, researchers were hoping to develop entirely automatic (i.e., completely computerized) expert systems. The expectations of people of what computers can do were frequently too idealistic. This situation radically changed after Richard M. Karp published his breakthrough paper: “Reducibility among Combinatorial Problems” in the early 1970s.[30] Thanks to Karp's work it became clear that there are certain limitations and possibilities when one designs computer algorithms. His findings describe what computers can do and what they cannot do. Many of the computational problems related to this type of expert systems have certain pragmatic limitations. These findings laid down the groundwork that led to the next developments in the field.[9]

In the 1990s and beyond, the term *expert system* and the idea of a standalone AI system mostly dropped from the IT lexicon. There are two interpretations of this. One is that "expert systems failed": the IT world moved on because expert systems did not deliver on their over hyped promise.[31][32] The other is the mirror opposite, that expert systems were simply victims of their success: as IT professionals grasped concepts such as rule engines, such tools migrated from being standalone tools for developing special purpose *expert* systems, to being one of many standard tools.[33] Many of the leading major business application suite vendors (such as SAP, Siebel, and Oracle) integrated expert system abilities into

their suite of products as a way of specifying business logic – rule engines are no longer simply for defining the rules an expert would use but for any type of complex, volatile, and critical business logic; they often go hand in hand with business process automation and integration environments.[34][35][36]

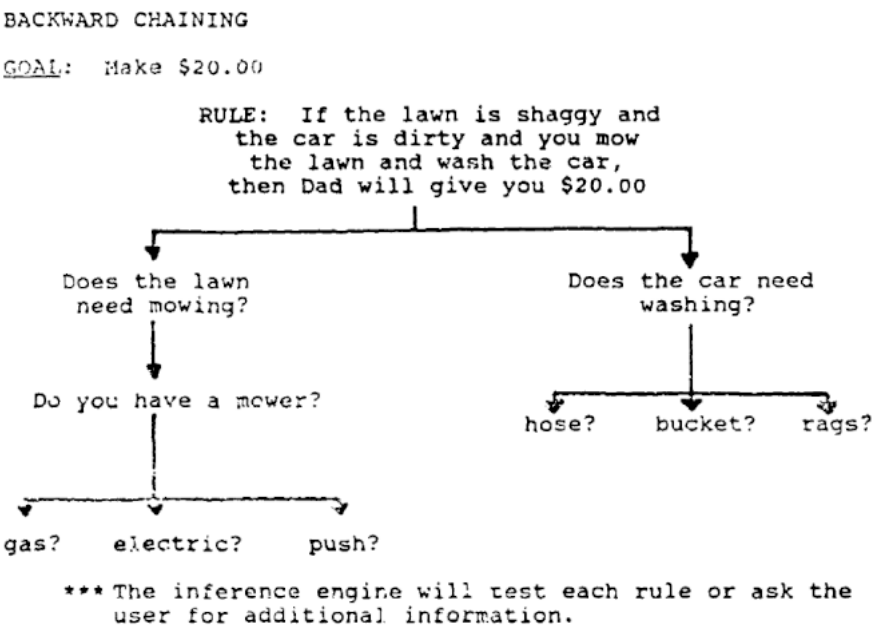
Current approaches to expert systems

The limitations of the previous type of expert systems have urged researchers to develop new types of approaches. They have developed more efficient, flexible, and powerful approaches in order to simulate the human decision-making process. Some of the approaches that researchers have developed are based on new methods of artificial intelligence (AI), and in particular in machine learning and data mining approaches with a feedback mechanism.[37] Recurrent neural networks often take advantage of such mechanisms. Related is the discussion on the disadvantages section.

Modern systems can incorporate new knowledge more easily and thus update themselves easily. Such systems can generalize from existing knowledge better and deal with vast amounts of complex data. Related is the subject of big data here. Sometimes these type of expert systems are called "intelligent systems."[9]

Software architecture

An expert system is an example of a knowledge-based system. Expert systems were the first commercial systems to use a knowledge-based architecture. In general view, an expert system includes the following components: a knowledge base, an inference engine, an explanation facility, a knowledge acquisition facility, and a user interface.[39] [40]



Illustrating example of *backward chaining* from a 1990 Master's Thesis[38]

The knowledge base represents facts about the world. In early expert systems such as Mycin and Dendral, these facts were represented mainly as flat assertions about variables. In later expert systems developed with commercial shells, the knowledge base took on more structure and used concepts from object-oriented programming. The world was represented as classes, subclasses, and instances and assertions were replaced by values of object instances. The rules worked by querying and asserting values of the objects.

The inference engine is an automated reasoning system that evaluates the current

state of the knowledge-base, applies relevant rules, and then asserts new knowledge into the knowledge base. The inference engine may also include abilities for explanation, so that it can explain to a user the chain of reasoning used to arrive at a particular conclusion by tracing back over the firing of rules that resulted in the assertion.[41]

There are mainly two modes for an inference engine: forward chaining and backward chaining. The different approaches are dictated by whether the inference engine is being driven by the antecedent (left hand side) or the consequent (right hand side) of the rule. In forward chaining an antecedent fires and asserts the consequent. For example, consider the following rule:

$$R1 : Man(x) \implies Mortal(x)$$

A simple example of forward chaining would be to assert Man(Socrates) to the system and then trigger the inference engine. It would match R1 and assert Mortal(Socrates) into the knowledge base.

Backward chaining is a bit less straight forward. In backward chaining the system looks at possible conclusions and works backward to see if they might be true. So if the system was trying to determine if Mortal(Socrates) is true it would find R1 and query the knowledge base to see if Man(Socrates) is true. One of the early innovations of expert systems shells was to integrate inference engines with a user interface. This could be especially powerful with backward chaining. If the system needs to know a particular fact but does not, then it can simply generate an input screen and ask the user if the information is known. So in this example, it could use R1 to ask the user if Socrates was a Man and then use that new information accordingly.

The use of rules to explicitly represent knowledge also enabled explanation abilities. In the simple example above if the system had used R1 to assert that Socrates was Mortal and a user wished to understand why Socrates was mortal they could query the system and the system would look back at the rules which fired to cause the assertion and present those rules to the user as an explanation. In English, if the user asked "Why is Socrates Mortal?" the system would reply "Because all men are mortal and Socrates is a man". A significant area for research was the generation of explanations from the knowledge base in natural English rather than simply by showing the more formal but less intuitive rules.[42]

As expert systems evolved, many new techniques were incorporated into various types of inference engines.[43] Some of the most important of these were:

- Truth maintenance. These systems record the dependencies in a knowledge-base so that when facts are altered, dependent knowledge can be altered accordingly. For example, if the system learns that Socrates is no longer known to be a man it will revoke the assertion that Socrates is mortal.
- Hypothetical reasoning. In this, the knowledge base can be divided up into many possible views, a.k.a. worlds. This allows the inference engine to explore multiple possibilities in parallel. For example, the system may want to explore the consequences of both assertions, what will be true if Socrates is a Man and what will be true if he is not?

- Uncertainty systems. One of the first extensions of simply using rules to represent knowledge was also to associate a probability with each rule. So, not to assert that Socrates is mortal, but to assert Socrates *may* be mortal with some probability value. Simple probabilities were extended in some systems with sophisticated mechanisms for uncertain reasoning, such as Fuzzy logic, and combination of probabilities.
- Ontology classification. With the addition of object classes to the knowledge base, a new type of reasoning was possible. Along with reasoning simply about object values, the system could also reason about object structures. In this simple example, Man can represent an object class and R1 can be redefined as a rule that defines the class of all men. These types of special purpose inference engines are termed classifiers. Although they were not highly used in expert systems, classifiers are very powerful for unstructured volatile domains, and are a key technology for the Internet and the emerging Semantic Web.[44][45]

Advantages

The goal of knowledge-based systems is to make the critical information required for the system to work explicit rather than implicit.[46] In a traditional computer program the logic is embedded in code that can typically only be reviewed by an IT specialist. With an expert system the goal was to specify the rules in a format that was intuitive and easily understood, reviewed, and even edited by domain experts rather than IT experts. The benefits of this explicit knowledge representation were rapid development and ease of maintenance.

Ease of maintenance is the most obvious benefit. This was achieved in two ways. First, by removing the need to write conventional code, many of the normal problems that can be caused by even small changes to a system could be avoided with expert systems. Essentially, the logical flow of the program (at least at the highest level) was simply a given for the system, simply invoke the inference engine. This also was a reason for the second benefit: rapid prototyping. With an expert system shell it was possible to enter a few rules and have a prototype developed in days rather than the months or year typically associated with complex IT projects.

A claim for expert system shells that was often made was that they removed the need for trained programmers and that experts could develop systems themselves. In reality, this was seldom if ever true. While the rules for an expert system were more comprehensible than typical computer code, they still had a formal syntax where a misplaced comma or other character could cause havoc as with any other computer language. Also, as expert systems moved from prototypes in the lab to deployment in the business world, issues of integration and maintenance became far more critical. Inevitably demands to integrate with, and take advantage of, large legacy databases and systems arose. To accomplish this, integration required the same skills as any other type of system.[47]

Summing up the benefits of using expert systems, the following can be highlighted: [39]

1. Increased availability and reliability: Expertise can be accessed on any computer hardware and the system always completes responses on time.
2. Multiple expertise: Several expert systems can be run simultaneously to solve a

problem. and gain a higher level of expertise than a human expert.
3. Explanation: Expert systems always describe of how the problem was solved.
4. Fast response: The expert systems are fast and able to solve a problem in real-time.
5. Reduced cost: The cost of expertise for each user is significantly reduced.

Disadvantages

The most common disadvantage cited for expert systems in the academic literature is the knowledge acquisition problem. Obtaining the time of domain experts for any software application is always difficult, but for expert systems it was especially difficult because the experts were by definition highly valued and in constant demand by the organization. As a result of this problem, a great deal of research in the later years of expert systems was focused on tools for knowledge acquisition, to help automate the process of designing, debugging, and maintaining rules defined by experts. However, when looking at the life-cycle of expert systems in actual use, other problems – essentially the same problems as those of any other large system – seem at least as critical as knowledge acquisition: integration, access to large databases, and performance.[48][49]

Performance could be especially problematic because early expert systems were built using tools (such as earlier Lisp versions) that interpreted code expressions without first compiling them. This provided a powerful development environment, but with the drawback that it was virtually impossible to match the efficiency of the fastest compiled languages (such as C). System and database integration were difficult for early expert systems because the tools were mostly in languages and platforms that were neither familiar to nor welcome in most corporate IT environments – programming languages such as Lisp and Prolog, and hardware platforms such as Lisp machines and personal computers. As a result, much effort in the later stages of expert system tool development was focused on integrating with legacy environments such as COBOL and large database systems, and on porting to more standard platforms. These issues were resolved mainly by the client–server paradigm shift, as PCs were gradually accepted in the IT environment as a legitimate platform for serious business system development and as affordable minicomputer servers provided the processing power needed for AI applications.[47]

Another major challenge of expert systems emerges when the size of the knowledge base increases. This causes the processing complexity to increase. For instance, when an expert system with 100 million rules was envisioned as the ultimate expert system, it became obvious that such system would be too complex and it would face too many computational problems.[50] An inference engine would have to be able to process huge numbers of rules to reach a decision.

How to verify that decision rules are consistent with each other is also a challenge when there are too many rules. Usually such problem leads to a satisfiability (SAT) formulation.[51] This is a well-known NP-complete problem Boolean satisfiability problem. If we assume only binary variables, say n of them, and then the corresponding search space is of size 2^n. Thus, the search space can grow exponentially.

There are also questions on how to prioritize the use of the rules in order to

operate more efficiently, or how to resolve ambiguities (for instance, if there are too many else-if sub-structures within a single rule) and so on.[52]

Other problems are related to the overfitting and overgeneralization effects when using known facts and trying to generalize to other cases not described explicitly in the knowledge base. Such problems exist with methods that employ machine learning approaches too.[53][54]

Another problem related to the knowledge base is how to make updates of its knowledge quickly and effectively.[55][56][57] Also how to add a new piece of knowledge (i.e., where to add it among many rules) is challenging. Modern approaches that rely on machine learning methods are easier in this regard.

Because of the above challenges, it became clear that new approaches to AI were required instead of rule-based technologies. These new approaches are based on the use of machine learning techniques, along with the use of feedback mechanisms.[9]

The key challenges that expert systems in medicine (if one considers computer-aided diagnostic systems as modern expert systems), and perhaps in other application domains, include issues related to aspects such as: big data, existing regulations, healthcare practice, various algorithmic issues, and system assessment.[58]

Finally, the following disadvantages of using expert systems can be summarized:[39]

1. Expert systems have superficial knowledge, and a simple task can potentially become computationally expensive.
2. Expert systems require knowledge engineers to input the data, data acquisition is very hard.
3. The expert system may choose the most inappropriate method for solving a particular problem.
4. Problems of ethics in the use of any form of AI are very relevant at present.
5. It is a closed world with specific knowledge, in which there is no deep perception of concepts and their interrelationships until an expert provides them.

Applications

Hayes-Roth divides expert systems applications into 10 categories illustrated in the following table. The example applications were not in the original Hayes-Roth table, and some of them arose well afterward. Any application that is not footnoted is described in the Hayes-Roth book.[41] Also, while these categories provide an intuitive framework to describe the space of expert systems applications, they are not rigid categories, and in some cases an application may show traits of more than one category.

Category	Problem addressed	Examples
Interpretation	Inferring situation descriptions from sensor data	Hearsay (speech recognition), PROSPECTOR
Prediction	Inferring likely consequences of given situations	Preterm Birth Risk Assessment[59]
Diagnosis	Inferring system malfunctions from observables	CADUCEUS, MYCIN, PUFF, Mistral,[60] Eydenet,[61] Kaleidos[62]
Design	Configuring objects under constraints	Dendral, Mortgage Loan Advisor, R1 (DEC VAX Configuration), SID (DEC VAX 9000 CPU)
Planning	Designing actions	Mission Planning for Autonomous Underwater Vehicle[63]
Monitoring	Comparing observations to plan vulnerabilities	REACTOR[64]
Debugging	Providing incremental solutions for complex problems	SAINT, MATHLAB, MACSYMA
Repair	Executing a plan to administer a prescribed remedy	Toxic Spill Crisis Management
Instruction	Diagnosing, assessing, and correcting student behaviour	SMH.PAL,[65] Intelligent Clinical Training,[66] STEAMER[67]
Control	Interpreting, predicting, repairing, and monitoring system behaviors	Real Time Process Control,[68] Space Shuttle Mission Control,[69] Smart Autoclave Cure of Composites[70]

Hearsay was an early attempt at solving voice recognition through an expert systems approach. For the most part this category of expert systems was not all that successful. Hearsay and all interpretation systems are essentially pattern recognition systems—looking for patterns in noisy data. In the case of Hearsay recognizing phonemes in an audio stream. Other early examples were analyzing sonar data to detect Russian submarines. These kinds of systems proved much more amenable to a neural network AI solution than a rule-based approach.

CADUCEUS and MYCIN were medical diagnosis systems. The user describes their symptoms to the computer as they would to a doctor and the computer returns a medical diagnosis.

Dendral was a tool to study hypothesis formation in the identification of organic molecules. The general problem it solved—designing a solution given a set of constraints—was one of the most successful areas for early expert systems applied to business domains such as salespeople configuring Digital Equipment Corporation (DEC) VAX computers and mortgage loan application development.

SMH.PAL is an expert system for the assessment of students with multiple disabilities.[65]

Mistral [60] is an expert system to monitor dam safety, developed in the 1990s by Ismes (Italy). It gets data from an automatic monitoring system and performs a diagnosis of the state of the dam. Its first copy, installed in 1992 on the Ridracoli Dam (Italy), is still operational 24/7/365. It has been installed on several dams in Italy and abroad (e.g., Itaipu Dam in Brazil), and on landslide sites under the name of Eydenet,[61] and on monuments under the name of Kaleidos.[62] Mistral is a

registered trade mark of CESI.

See also

- AI winter
- CLIPS
- Constraint logic programming
- Constraint satisfaction
- Knowledge engineering
- Learning classifier system
- Rule-based machine learning

References

1. Jackson, Peter (1998). *Introduction To Expert Systems* (3 ed.). Addison Wesley. p. 2. ISBN 978-0-201-87686-4.
2. "Conventional programming" (https://web.archive.org/web/20121014124656/http://www.pcmag.com/encyclopedia_term/0%2C2542%2Ct%3Dconventional+programming%26i%3D40325%2C00.asp). Pcmag.com. Archived from the original (https://www.pcmag.com/encyclopedia_term/0,2542,t=conventional+programming&i=40325,00.asp) on 2012-10-14. Retrieved 2013-09-15.
3. Leondes, Cornelius T. (2002). *Expert systems: the technology of knowledge management and decision making for the 21st century*. pp. 1–22. ISBN 978-0-12-443880-4.
4. Russell, Stuart; Norvig, Peter (1995). *Artificial Intelligence: A Modern Approach* (https://web.archive.org/web/20140505045226/http://stpk.cs.rtu.lv/sites/all/files/stpk/materiali/MI/Artificial%20Intelligence%20A%20Modern%20Approach.pdf) (PDF). Simon & Schuster. pp. 22–23. ISBN 978-0-13-103805-9. Archived from the original (http://stpk.cs.rtu.lv/sites/all/files/stpk/materiali/MI/Artificial%20Intelligence%20A%20Modern%20Approach.pdf) (PDF) on 5 May 2014. Retrieved 14 June 2014.
5. Luger & Stubblefield 2004, pp. 227–331.
6. Nilsson 1998, chpt. 17.4.
7. McCorduck 2004, pp. 327–335, 434–435.
8. Crevier 1993, pp. 145–62, 197−203.
9. Yanase J, Triantaphyllou E (2019). "A Systematic Survey of Computer-Aided Diagnosis in Medicine: Past and Present Developments". *Expert Systems with Applications*. **138**: 112821. doi:10.1016/j.eswa.2019.112821 (https://doi.org/10.1016%2Fj.eswa.2019.112821). S2CID 199019309 (https://api.semanticscholar.org/CorpusID:199019309).
10. Ledley RS, and Lusted LB (1959). "Reasoning foundations of medical diagnosis". *Science*. **130** (3366): 9–21. Bibcode:1959Sci...130....9L (https://ui.adsabs.harvard.edu/abs/1959Sci...130....9L). doi:10.1126/science.130.3366.9 (https://doi.org/10.1126%2Fscience.130.3366.9). PMID 13668531 (https://pubmed.ncbi.nlm.nih.gov/13668531).
11. Weiss SM, Kulikowski CA, Amarel S, Safir A (1978). "A model-based method for computer-aided medical decision-making". *Artificial Intelligence*. **11** (1–2): 145–172. doi:10.1016/0004-3702(78)90015-2 (https://doi.org/10.1016%2F0004-3702%2878%2990015-2).

12. Schwartz WB (1970). "Medicine and the computer: the promise and problems of change". *New England Journal of Medicine*. **283** (23): 1257–1264. doi:10.1056/NEJM197012032832305 (https://doi.org/10.1056%2FNEJM197012032832305). PMID 4920342 (https://pubmed.ncbi.nlm.nih.gov/4920342).
13. Bleich HL (1972). "Computer-based consultation: Electrolyte and acid-base disorders". *The American Journal of Medicine*. **53** (3): 285–291. doi:10.1016/0002-9343(72)90170-2 (https://doi.org/10.1016%2F0002-9343%2872%2990170-2). PMID 4559984 (https://pubmed.ncbi.nlm.nih.gov/4559984).
14. Rosati RA, McNeer JF, Starmer CF, Mittler BS, Morris JJ, and Wallace AG (1975). "A new information system for medical practice". *Archives of Internal Medicine*. **135** (8): 1017–1024. doi:10.1001/archinte.1975.00330080019003 (https://doi.org/10.1001%2Farchinte.1975.00330080019003). PMID 1156062 (https://pubmed.ncbi.nlm.nih.gov/1156062).
15. Gorry GA, Kassirer JP, Essig A, and Schwartz WB (1973). "Decision analysis as the basis for computer-aided management of acute renal failure". *The American Journal of Medicine*. **55** (4): 473–484. doi:10.1016/0002-9343(73)90204-0 (https://doi.org/10.1016%2F0002-9343%2873%2990204-0). PMID 4582702 (https://pubmed.ncbi.nlm.nih.gov/4582702).
16. Szolovits P, Patil RS, and Schwartz WB (1988). "Artificial intelligence in medical diagnosis". *Annals of Internal Medicine*. **108** (1): 80–87. doi:10.7326/0003-4819-108-1-80 (https://doi.org/10.7326%2F0003-4819-108-1-80). PMID 3276267 (https://pubmed.ncbi.nlm.nih.gov/3276267).
17. Shortliffe EH, and Buchanan BG (1975). "A model of inexact reasoning in medicine". *Mathematical Biosciences*. **23** (3–4): 351–379. doi:10.1016/0025-5564(75)90047-4 (https://doi.org/10.1016%2F0025-5564%2875%2990047-4).
18. Miller RA, Pople Jr HE, and Myers JD (1982). "Internist-I, an experimental computer-based diagnostic consultant for general internal medicine". *New England Journal of Medicine*. **307** (8): 468–476. doi:10.1056/NEJM198208193070803 (https://doi.org/10.1056%2FNEJM198208193070803). PMID 7048091 (https://pubmed.ncbi.nlm.nih.gov/7048091).
19. Feigenbaum, Edward; McCorduck, Pamela (1984). *The fifth generation*. Addison-Wesley. pp. 1–275. ISBN 978-0451152640.
20. kenyon.edu: AI Timeline (http://biology.kenyon.edu/slonc/bio3/AI/TIMELINE/timeline.html), retrieved October 27, 2018
21. Edward Feigenbaum, 1977. Paraphrased by Hayes-Roth, et al.
22. Hayes-Roth, Frederick; Waterman, Donald; Lenat, Douglas (1983). *Building Expert Systems* (https://archive.org/details/buildingexpertsy00temd/page/6). Addison-Wesley. pp. 6–7 (https://archive.org/details/buildingexpertsy00temd/page/6). ISBN 978-0-201-10686-2.
23. George F. Luger and William A. Stubblefield, Benjamin/Cummings Publishers, Rule-Based Expert System Shell: example of code using the Prolog rule-based expert system shell
24. A. Michiels (http://promethee.philo.ulg.ac.be/engdep1/download/prolog/htm_docs/prolog.htm) Archived (https://web.archive.org/web/20120402132354/http://promethee.philo.ulg.ac.be/engdep1/download/prolog/htm_docs/prolog.htm) 2012-04-02 at the Wayback Machine, Université de Liège, Belgique: "PROLOG, the first declarative language
25. Durkin, J. Expert Systems: Catalog of Applications. Intelligent Computer Systems, Inc., Akron, OH, 1993.

26. Orfali, Robert (1996). *The Essential Client/Server Survival Guide* (https://archive.org/details/essentialclients00orfa/page/1). New York: Wiley Computer Publishing. pp. 1–10 (https://archive.org/details/essentialclients00orfa/page/1). ISBN 978-0-471-15325-2.
27. Hurwitz, Judith (2011). *Smart or Lucky: How Technology Leaders Turn Chance into Success* (https://books.google.com/books?id=3KrTQzQHl7AC&q=expert+systems+failed+to+live+up+to+hype&pg=PA164). John Wiley & Son. p. 164. ISBN 978-1118033784. Retrieved 29 November 2013.
28. Dunn, Robert J. (September 30, 1985). "Expandable Expertise for Everyday Users" (https://books.google.com/books?id=iS8EAAAAMBAJ&pg=PA30). *InfoWorld*. **7** (39): 30. Retrieved 2011-03-13.
29. Carl S. Gibson, et al, VAX 9000 SERIES, Digital Technical Journal of Digital Equipment Corporation, Volume 2, Number 4, Fall 1990, pp118-129.
30. Richard M. Karp (1972). "Reducibility Among Combinatorial Problems" (http://www.cs.berkeley.edu/~luca/cs172/karp.pdf) (PDF). In R. E. Miller; J. W. Thatcher (eds.). *Complexity of Computer Computations*. New York: Plenum. pp. 85–103.
31. "AI Expert Newsletter: W is for Winter" (https://web.archive.org/web/20131109201636/http://www.ainewsletter.com/newsletters/aix_0501.htm#w). Archived from the original (http://www.ainewsletter.com/newsletters/aix_0501.htm#w) on 2013-11-09. Retrieved 2013-11-29.
32. "Leith P., "The rise and fall of the legal expert system", in European Journal of Law and Technology, Vol 1, Issue 1, 2010" (http://ejlt.org//article/view/14/1).
33. Haskin, David (January 16, 2003). "Years After Hype, 'Expert Systems' Paying Off For Some" (http://www.datamation.com/netsys/article.php/1570851/Years-After-Hype-Expert-Systems-Paying-Off-For-Some.htm). *Datamation*. Retrieved 29 November 2013.
34. SAP News Desk. "SAP News Desk IntelliCorp Announces Participation in SAP EcoHub" (http://laszlo.sys-con.com/node/946452). *laszlo.sys-con.com*. LaszloTrack. Retrieved 29 November 2013.
35. Pegasystems. "Smart BPM Requires Smart Business Rules" (http://www.pega.com/business-rules). *pega.com*. Retrieved 29 November 2013.
36. Zhao, Kai; Ying, Shi; Zhang, Linlin; Hu, Luokai (9–10 Oct 2010). "Achieving business process and business rules integration using SPL". *Future Information Technology and Management Engineering (FITME)*. Vol. 2. Changzhou, China: IEEE. pp. 329–332. doi:10.1109/fitme.2010.5656297 (https://doi.org/10.1109%2Ffitme.2010.5656297). ISBN 978-1-4244-9087-5.
37. Chung, Junyoung; Gulcehre, Caglar; Cho, Kyunghyun; Bengio, Yoshua (2015-06-01). "Gated Feedback Recurrent Neural Networks" (http://proceedings.mlr.press/v37/chung15.html). *International Conference on Machine Learning*. PMLR: 2067–2075. arXiv:1502.02367 (https://arxiv.org/abs/1502.02367).
38. David C. England (Jun 1990). *An Expert System for the Management of Hazardous Materials at a Naval Supply Center* (https://commons.wikimedia.org/wiki/File:An_expert_system_for_the_management_of_hazardous_materials_at_a_Naval_Supply_Center_(IA_anexpertsystemfo1094530640).pdf) (PDF) (Master's thesis). Naval Postgraduate School Monterey/CA. Here: p.21.
39. Kiryanov, Denis Aleksandrovich (2021-12-21). "Hybrid categorical expert system for use in content aggregation" (https://dx.doi.org/10.7256/2454-0714.2021.4.37019). *Software Systems and Computational Methods* (4): 1–22. doi:10.7256/2454-0714.2021.4.37019 (https://doi.org/10.7256%2F2454-0714.2021.4.37019). ISSN 2454-0714 (https://www.worldcat.org/issn/2454-0714). S2CID 245498498 (https://api.semanticscholar.org/CorpusID:245498498).

40. Smith, Reid (May 8, 1985). "Knowledge-Based Systems Concepts, Techniques, Examples" (http://www.reidgsmith.com/Knowledge-Based_Systems_-_Concepts_Techniques_Examples_08-May-1985.pdf) (PDF). *Reid G. Smith*. Retrieved 9 November 2013.
41. Hayes-Roth, Frederick; Waterman, Donald; Lenat, Douglas (1983). *Building Expert Systems* (https://archive.org/details/buildingexpertsy00temd). Addison-Wesley. ISBN 978-0-201-10686-2.
42. Nabil Arman (http://www.ccis2k.org/iajit/PDF/vol.4,no.1/9-Nabil.pdf), Polytechnic University of Palestine, January 2007, Fault Detection in Dynamic Rule Bases Using Spanning Trees and Disjoin Sets: ""
43. Mettrey, William (1987). "An Assessment of Tools for Building Large Knowledge-Based Systems" (https://web.archive.org/web/20131110022104/http://www.aaai.org/ojs/index.php/aimagazine/article/viewArticle/625). *AI Magazine*. **8** (4). Archived from the original (http://www.aaai.org/ojs/index.php/aimagazine/article/viewArticle/625) on 2013-11-10. Retrieved 2013-11-29.
44. MacGregor, Robert (June 1991). "Using a description classifier to enhance knowledge representation". *IEEE Expert*. **6** (3): 41–46. doi:10.1109/64.87683 (https://doi.org/10.1109%2F64.87683). S2CID 29575443 (https://api.semanticscholar.org/CorpusID:29575443).
45. Berners-Lee, Tim; Hendler, James; Lassila, Ora (May 17, 2001). "The Semantic Web A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities" (https://web.archive.org/web/20130424071228/http://www.cs.umd.edu/~golbeck/LBSC690/SemanticWeb.html). *Scientific American*. **284** (5): 34–43. doi:10.1038/scientificamerican0501-34 (https://doi.org/10.1038%2Fscientificamerican0501-34). Archived from the original (http://www.cs.umd.edu/~golbeck/LBSC690/SemanticWeb.html) on April 24, 2013.
46. Hayes-Roth, Frederick; Waterman, Donald; Lenat, Douglas (1983). *Building Expert Systems* (https://archive.org/details/buildingexpertsy00temd/page/6). Addison-Wesley. p. 6 (https://archive.org/details/buildingexpertsy00temd/page/6). ISBN 978-0-201-10686-2.
47. Wong, Bo K.; Monaco, John A.; Monaco (September 1995). "Expert system applications in business: a review and analysis of the literature" (http://dl.acm.org/citation.cfm?id=218565&CFID=383824649&CFTOKEN=44575196). *Information and Management*. **29** (3): 141–152. doi:10.1016/0378-7206(95)00023-p (https://doi.org/10.1016%2F0378-7206%2895%2900023-p). Retrieved 29 November 2013.
48. Kendal, S.L.; Creen, M. (2007). *An introduction to knowledge engineering*. London: Springer. ISBN 978-1-84628-475-5. OCLC 70987401 (https://www.worldcat.org/oclc/70987401).
49. Feigenbaum, Edward A.; McCorduck, Pamela (1983). *The fifth generation* (1st ed.). Reading, MA: Addison-Wesley. ISBN 978-0-201-11519-2. OCLC 9324691 (https://www.worldcat.org/oclc/9324691).
50. Douglas B. Lenat (1992). "On the thresholds of knowledge". In David Kirsh (ed.). *Foundations Of Artificial Intelligence*. MIT Press. pp. 185–250.
51. Bezem M (1988). "Consistency of rule-based expert systems" (https://ir.cwi.nl/pub/6113). *In International Conference on Automated Deduction*. Lecture Notes in Computer Science. **310**: 151–161. doi:10.1007/BFb0012830 (https://doi.org/10.1007%2FBFb0012830). ISBN 3-540-19343-X.
52. Mak B, Schmitt BH, and Lyytinen K (1997). "User participation in knowledge update of expert systems". *Information & Management*. **32** (2): 55–63. doi:10.1016/S0378-7206(96)00010-9 (https://doi.org/10.1016%2FS0378-7206%2896%2900010-9).

53. Pham HN, Triantaphyllou E (2008). "The Impact of Overfitting and Overgeneralization on the Classification Accuracy in Data Mining". *Soft Computing for Knowledge Discovery and Data Mining*: 391–431.
54. Pham HN, Triantaphyllou E (2008). "Prediction of diabetes by employing a new data mining approach which balances fitting and generalization". *Computer and Inf. Science G*: 11–26.
55. Shan N, and Ziarko W (1995). "Data-based acquisition and incremental modification of classification rules". *Computational Intelligence*. **11** (2): 357–370. doi:10.1111/j.1467-8640.1995.tb00038.x (https://doi.org/10.1111%2Fj.1467-8640.1995.tb00038.x). S2CID 38974914 (https://api.semanticscholar.org/CorpusID:38974914).
56. Coats PK (1988). "Why expert systems fail". *Financial Management*. **17** (3): 77–86. doi:10.2307/3666074 (https://doi.org/10.2307%2F3666074). JSTOR 3666074 (https://www.jstor.org/stable/3666074).
57. Hendriks PH, and Vriens DJ (1999). "Knowledge-based systems and knowledge management: friends or foes?". *Information & Management*. **35** (2): 113–125. doi:10.1016/S0378-7206(98)00080-9 (https://doi.org/10.1016%2FS0378-7206%2898%2900080-9).
58. Yanase J, Triantaphyllou E (2019). "The Seven Key Challenges for the Future of Computer-Aided Diagnosis in Medicine". *International Journal of Medical Informatics*. **129**: 413–422. doi:10.1016/j.ijmedinf.2019.06.017 (https://doi.org/10.1016%2Fj.ijmedinf.2019.06.017). PMID 31445285 (https://pubmed.ncbi.nlm.nih.gov/31445285). S2CID 198287435 (https://api.semanticscholar.org/CorpusID:198287435).
59. Woolery, L.K.; Grzymala-Busse, J (1994). "Machine learning for an expert system to predict preterm birth risk" (https://www.ncbi.nlm.nih.gov/pmc/articles/PMC116227). *Journal of the American Medical Informatics Association*. **1** (6): 439–446. doi:10.1136/jamia.1994.95153433 (https://doi.org/10.1136%2Fjamia.1994.95153433). PMC 116227 (https://www.ncbi.nlm.nih.gov/pmc/articles/PMC116227). PMID 7850569 (https://pubmed.ncbi.nlm.nih.gov/7850569).
60. Salvaneschi, Paolo; Cadei, Mauro; Lazzari, Marco (1996). "Applying AI to structural safety monitoring and evaluation" (http://www.computer.org/csdl/mags/ex/1996/04/x4024-abs.html). *IEEE Expert*. **11** (4): 24–34. doi:10.1109/64.511774 (https://doi.org/10.1109%2F64.511774). Retrieved 5 March 2014.
61. Lazzari, Marco; Salvaneschi, Paolo (1999). "Embedding a geographic information system in a decision support system for landslide hazard monitoring" (http://dinamico2.unibg.it/lazzari/doc/embedding-authors-copy.pdf) (PDF). *International Journal of Natural Hazards*. **20** (2–3): 185–195. doi:10.1023/A:1008187024768 (https://doi.org/10.1023%2FA%3A1008187024768). S2CID 1746570 (https://api.semanticscholar.org/CorpusID:1746570).
62. Lancini, Stefano; Lazzari, Marco; Masera, Alberto; Salvaneschi, Paolo (1997). "Diagnosing Ancient Monuments with Expert Software" (http://dinamico2.unibg.it/lazzari/doc/structural-engineering-authors-copy.pdf) (PDF). *Structural Engineering International*. **7** (4): 288–291. doi:10.2749/101686697780494392 (https://doi.org/10.2749%2F101686697780494392).
63. Kwak, S.. H. (1990). "A mission planning expert system for an autonomous underwater vehicle". *Proceedings of the 1990 Symposium on Autonomous Underwater Vehicle Technology*: 123–128. doi:10.1109/AUV.1990.110446 (https://doi.org/10.1109%2FAUV.1990.110446). S2CID 60476847 (https://api.semanticscholar.org/CorpusID:60476847).

64. Nelson, W. R. (1982). "REACTOR: An Expert System for Diagnosis and Treatment of Nuclear Reactors".
65. Hofmeister, Alan (1994). "SMH.PAL: an expert system for identifying treatment procedures for students with severe disabilities" (https://web.archive.org/web/20131203033204/http://www.freepatentsonline.com/article/Exceptional-Children/15824013.html). *Exceptional Children*. **61** (2). Archived from the original (http://www.freepatentsonline.com/article/Exceptional-Children/15824013.html) on 3 December 2013. Retrieved 30 November 2013.
66. Haddawy, P; Suebnukarn, S. (2010). "Intelligent Clinical Training Systems". *Methods Inf Med*. **49** (4): 388–9. CiteSeerX 10.1.1.172.60 (https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.172.60). doi:10.1055/s-0038-1625342 (https://doi.org/10.1055%2Fs-0038-1625342). PMID 20686730 (https://pubmed.ncbi.nlm.nih.gov/20686730).
67. Hollan, J.; Hutchins, E.; Weitzman, L. (1984). "STEAMER: An interactive inspectable simulation-based training system". *AI Magazine*.
68. Stanley, G.M. (July 15–17, 1991). "Experience Using Knowledge-Based Reasoning in Real Time Process Control" (http://www.gregstanleyandassociates.com/whitepapers/IFAC91objectPaper.pdf) (PDF). *Plenary Paper Presented at: International Federation of Automatic Control (IFAC) Symposium on Compute R Aided Design in Control Systems*. Retrieved 3 December 2013.
69. Rasmussen, Arthur; Muratore, John F.; Heindel, Troy A. (February 1990). "The INCO Expert System Project: CLIPS in Shuttle mission control" (https://www.researchgate.net/publication/4702412). *NTRS*. Retrieved 30 November 2013.
70. Ciriscioli, P. R., and G. S. Springer (1990). *"Smart Autoclave Cure of Composites"*. ISBN 9781003209010.

External links

- Artificial Intelligence (https://curlie.org/Computers/Artificial_Intelligence/) at Curlie
- Expert System tutorial on Code Project (http://www.codeproject.com/KB/recipes/ArtificialAdvice-1.aspx)