EXPERT SYSTEMS AND KNOWLEDGE ACQUISITION

Roberto Melli
University of Rome 1 "La Sapienza", Italy

Keywords: Expert Systems, Knowledge-Based Systems, Artificial Intelligence, Knowledge Acquisition

Contents

Summary

Expert Systems are examples of particular applications of the techniques of Artificial Intelligence, aimed at the construction of "codes" (more properly, "automatic procedures") capable of logically selective rather than deterministic behavior. While the heart of such systems is their Inference Engine (i.e., the portion in which the rules are properly compiled and consulted to extract higher level information on the System's future actions), their indispensable "fuel" is a certain amount of knowledge about the universe they are supposed to interact with. Such knowledge must be properly extracted, catalogued, possibly classified, logically collected or refragmented, and systematically fed into a special data base called the Knowledge Base of the ES. This chapter describes the principles of Knowledge acquisition and Knowledge Base assemblage, and discusses some of the problems that may arise with the collection and classification of knowledge.

1. Introduction

Scope of any Knowledge-based (KB) method applied to an engineering task is that of assisting -or substituting- the "expert" engineer in performing tasks which can be suitably represented in a form which allows for knowledge manipulation on the part of a computer code, called the Expert System (ES) (see *Artificial Intelligence and Energy Systems: Scope and Definition*). This "knowledge", in any way we wish to define it, must be acquired, catalogued and somehow manipulated before automatic inference engines can use it. The initial treatment of knowledge is so important that a new professional figure has emerged in the last ten or twenty years, namely the *Knowledge Engineer*: this is a person not necessarily possessing either domain or with Artificial Intelligence (AI) coding notions, who must "scout" for all the relevant and pertinent information bits of interest for the specific application, and repackage the collected knowledge in such a way as to make it useful for an AI-code. Therefore, long before launching a coding activity, the knowledge engineer should consider the following questions:

- How can the knowledge necessary for expert-level performance be represented as symbolic data structures for computer use?
- How can one achieve flexibility in adding and changing knowledge in the development of a *Knowledge Base*?
- How can the knowledge in the specific field be systematically acquired?
- Can such knowledge be discovered by an autonomously acting program?
- What designs are available for the inference procedure to be used by the intelligent artifact?
- How can one achieve efficient and accurate performance as the extension of the problem space increases?

If the power of an AI program is primarily a function of the quality and completeness of its knowledge base, then these are the critical bottlenecks of Knowledge-based Systems (KBS). Four basic principles guide the Knowledge Engineer in his knowledge gathering activity:

- *The KNOWLEDGE PRINCIPLE*: "a system exhibits intelligent understanding and action at a high level of competence primarily because of the specific knowledge it can be brought to bear: the concepts, facts, representations, methods, models and heuristics about its problem domain".
- *The WELL-FORMEDNESS THRESHOLD*: "for each task, there is some minimum knowledge needed for one to even formulate it".
- *The COMPETENCE THRESHOLD*: "difficult tasks succumb nonlinearly to knowledge. There is an ever-greater payoff (convenience) to adding each bit of knowledge, up to some level of competence. Beyond that, additional knowledge is useful but not frequently needed (e.g., it is used to handle exceptions or anomalies). When a point is reached in which almost all of these exceptions are handled as well, the addition of knowledge carries very little usefulness, and should be avoided".
- *The INCREASING BREADTH/LIMITING DEPTH PRINCIPLE*: "intelligent performance often requires problem solvers to ascend the conceptual ladder towards increasingly general knowledge, and/or to draw analogies to specific knowledge from seemingly distant domains. When this happens, and some unexplained logical or physical similarity can be detected between two facts A and B pertaining to distant domains, it is more useful to put these similarities to immediate use than to try to analyze in depth the logical paths which lead to these similarities".

This chapter deals with some of the available knowledge *acquisition* and *representation* methods.

2. General Knowledge Representation for Design Purposes

Selecting a suitable representation for the domain knowledge is one of the first problems encountered when building a KBS. There are some general principles that should guide this representation, though there is a considerable degree of disagreement among specialists in the field. The views presented here are therefore to be taken as a (not necessarily unbiased!) generalization, and ought to be complemented with other views and approaches (see the references in *Artificial Intelligence and Energy Systems: Scope and Definition*).

- "Real" knowledge is hierarchical: so should its representation be;
- The (common) language which is originally used to describe a specific piece of knowledge ought to be translated into the KB without forcing it into a more precise frame than that which was originally its own: for example, if the original information is sketchy or vague or dubious, care should be exercised to retain some sort of approximation, vagueness or uncertainty in the relevant KB objects;
- It is advisable to write separate production rules for each level of the knowledge hierarchy: these rules may and may not be the same at different levels;
- Since in most engineering cases the Knowledge Base is changing in time, it is advisable to use a flexible domain-knowledge representation, consisting in a "network" of cross-referencing rules. "Tabular" knowledge representations are quite rigid and ought to be avoided.

In practice, these guidelines are not difficult to implement, and their consistent application constitutes already a substantial step towards the successful compilation of an Expert System (ES). They result in general “procedural rules” which form a sort of “syntax” of the implementation language: the “grammar” of the language consists of a set of types under which the available knowledge can be catalogued. The list here below is neither complete nor absolute, and it has been derived from only a portion of the published applications; but it constitutes a rather complete “dictionary” of knowledge classification that covers most Thermal Systems Design problems. According to this list, both the conceptual (qualitative) and the physical (quantitative) knowledge needed for the implementation of an Expert Assistant for Thermal Systems Synthesis fall under the following types:

- *Objects*: components, fluids, controls and equipment in general, human operators;
- *Properties*: attributes of an object. They can be qualitative (“component A is more apt than component B to the type of service described by design specification rule R”) or quantitative (“maximum gas turbine inlet temperature T_{max} = 1300K”), or both (“Deaerator extraction pump should be a booster for the main feedwater pump AND its net head ought to be higher than 0.02 MPa”);
- *Relationships*: (virtual) actions that directly or indirectly connect components or processes. They can be physical (“Boiler needs air, fuel and preheated water as inputs”) or logical (“If the fuel is coal, then a check of its sulfur content is

necessary to decide whether to consider a desulphurization unit or not"). Relationships can be further subdivided into:

1. 3.1 - *Time relationships*: relationships containing time (in the physical sense) as an independent variable. Example: "The availability of a sufficient mass flow rate of external cooling water must be ascertained *before* considering the possibility of inserting a water-cooled condenser in a fossil-fuelled power plant".
2. 3.2 - *Cost relationships*: relationships that have a cost measure (monetary, exergetic, resource-based, etc.) as an independent variable. Example: "The exergetic cost of the unit mass of natural gas is higher than the exergetic cost of the unit mass of pre-treated urban refuse".
3. 3.3 - *Causal relationships:* of the type *"if p then q"*.

- *Aggregations*: super- or subsets of a system. For example, a plant based on a combined cycle has both the steam plant and the turbogas as its subsets; in turn, both the turbogas and the steam plant possess their components as subsets: in this case, we would call "aggregation level 0" the level at which the single components are considered and described; "aggregation level 1" the level at which the turbogas and the steam plant are considered, and "aggregation level 2" the level at which the entire process is the "black box".
- *Engineering assumptions*: all rules, specifications, descriptions, principles, assumptions and "rules-of-thumb" which derive from the direct experience of process- or design engineers. They include all the heuristics usually applied in the specific design activity.
- *Procedures*: norms, laws, regulations etc. It is useful to include in this type also all "approximate", "vague" or "uncertain" descriptions of operative procedures ("If excessive vibrations are detected on a compressor shaft, stall-check procedure must be launched").
- *Analogies*: all types of analogical reasoning known to apply to the field of application of the ES.
- *Behavioral rules*: qualitative or quantitative information available on the operation of any component of process of interest. It is advisable that these rules also include approximate and uncertain behavior.
- *Approximate quantitative analysis*: "bulk" estimates or first-order approximations to process- or component calculations. Example: "If a gas turbine blade row is air-cooled, there would be no need of special alloys up to gas temperatures of about 1100K"; or "Rough energy balance of the combustion chamber is: $m_{air,in}*h_{air,in} + m_{fuel} * LHV_{fuel} = m_{gas}*c_{p,gas} *T_{gas}$". In this last case, a series of additional information bits are needed to qualify $m_{air,in}$ as the incoming air mass flow rate, $h_{air,in}$ as its enthalpy, m_{fuel} as the fuel mass flow rate, LHV_{fuel} as its lower heating value, m_{gas} as the sum of m_{air} and m_{fuel}, $c_{p,gas}$ as the average specific heat of the mixture, and T_{gas} as the reactants temperature.

Using this grammar, and including in the paradigms of the ES to be constructed the syntax described by rules I-IV above, a sufficiently detailed, well-organized, easily accessible knowledge representation is easily achieved.

3. The Knowledge Acquisition Problem

3.1. Acquisition of Knowledge is a formidable Problem in itself

The so-called *Feigenbaum bottleneck problem*, formulated in 1977, stated that “Expert knowledge acquisition is a problematic bottleneck in the construction of application-oriented Expert Systems”. After 25 years, it is still true that a good method for constructing and maintaining (updating) the KB is the best guarantee of success of an ES. A substantial portion of the bottleneck problem lies in the wrong use that is made of expert’s advice: often, it is required of the Domain Expert to systematically analyze all implications of a certain design decision, thus completely ignoring a crucial fact, namely, that the important content of the expert’s Knowledge is his application experience, and not system theory.

Therefore, in this sense, this part of the bottleneck is caused by a wrong approach on the part of the Knowledge Engineer, and can be easily removed. Nevertheless, the problem is not only found in *collecting* domain-knowledge (though this can be difficult in some specific cases), but in *systematically storing it in the KB*. This “storage” can be best organized with the help of specifically designed knowledge-acquisition computer codes, which "guide" the user through the process by making a large body of meta-level knowledge-acquisition rules accessible to him: unfortunately, general KB-acquisition codes are not readily available, since ES-assisted KB acquisition has been shown to work for rather small problems, but becomes unwieldy as the complexity of the domain increases.

This "expert knowledge acquisition" can be performed in two "modes": *automatically* or *interactively*. In the interactive mode, the Knowledge Engineer uses his specific expertise in knowledge structures to help the designer fill out certain structures with portions of the available domain-specific knowledge. The general structures of knowledge remain the same from one application domain to another, but they are filled with different concepts and data (different objects) according to the specific problem under consideration.

The representation is bound to be closely tied to conditional rules, and a significant degree of code/user interaction is needed. In the automatic mode, the Knowledge Engineer is given the domain-knowledge in "chunks", i.e., several pieces of information at a time, and works on them independently from the user, trying to "induce" from these primitive facts the structure of the knowledge it will have to use. Adding other "chunks" helps him verify his conclusions, to amend them as necessary, and to generalize further the induced knowledge structure.

-
-
-

Bibliography

De Marco M. et al. (1993). COLOMBO: an expert system for process design of thermal power plants. *ASME-AES* vol.1/10. [The first publication describing the package that was to become a paradigm for a series of process synthesizers by the same authors]

Ertas A. and Jones J. C. (1996). *The Engineering Design Process*. J. Wiley & Sons. [A systematization of some engineering design procedures.]

Giarratano J. and Riley G. (1989). *Expert Systems: Principles and Programming*. Boston: PWS Publishing Co. [A valuable source of modular procedures for object-oriented, knowledge based programming.]

Hori M. et al.: Configuring problem-solving methods: a CAKE perspective. *Knowl. Acquis.*, vol.6, no.4, 1994, pp. 461-488. [An example of integration into an Expert System of a “library” of reusable knowledge. “CAKE” means “Computer-Aided Knowledge Engineering”]

Lloyd J. W., ed. (1990). *Computational Logic*. Springer Verlag Basic Res. Series. [A collection of papers on various topics related to the translation into computer language of the principles of relational logic.]

Maiorano M. et al. (2002). An Expert Assistant For The Automatic Synthesis Of Heat Exchanger Networks. *J. Energy Research Technology* 132 (2). [An improved version of an Expert Design Assistant (EDA) capable of performing a synthesis and a second-Law based optimization of heat exchanger networks. A large series of benchmarks show that this EDA performs consistently as well as both pinch- and heuristic methods.]

Newell A. (1990). *Unified theories of cognition*. Harvard University Press. [A thorough critical explanation of the principles of cognitive science.]

Pearl J. (1984). *Heuristics*. Addison-Wesley. [A remarkable treatise that explains the details of a set of procedures that go well beyond the “trial and error” formulation.]

Sciubba E. and Melli R. (1998). *Artificial Intelligence in Thermal Systems Design: Concepts and Applications*. New York: NOVA SCIENCE Publishers. [An introductory but complete textbook on AI topics, very strongly biased towards applications. A textbook for engineering-oriented AI users.]

Shiffmann W. H. and Geffers H. W. (1993). Adaptive control of dynamic systems by back-propagation networks. *Neural Networks* **6**(4). [An interesting application of artificial neural networks, with many detailed theoretical explanations.]

Sommer E. et al.: What online machine learning can do for Knowledge Acquisition – A case study, *Knowledge Acquisition*, vol.6, no.4, 1994, pp. 435-460. [A step-by-step account of the construction of an Expert System for Knowledge Acquisition]

Sriram R. D. (1997). *Intelligent Systems for Engineering*. Springer Verlag. [A complete, well written, well balanced textbook on AI. For the quantity and quality of information contained, it constitutes a necessary reference for workers in this area.]

Biographical Sketch

Roberto Melli is a Researcher at the Department of Mechanical and Aeronautical Engineering of the University of Roma 1 “La Sapienza” (UDR1), Roma, Italy. He received a Master in Mechanical Engineering from UDR1 in 1974. From 1974 to 75, he was a Research Assistant at the Chair of Machines in the same University. As a faculty member he lectures on Machinery and Energy Systems Design in Nuclear Engineering Master level courses.

His research activities are equally divided in two main fields:

1. Energy Systems Simulation and Design

2. Applications of AI-related techniques and procedures to the Design, Synthesis and Optimization of Energy Systems

His publications include more than thirty journal articles (mostly on international refereed Journals in the field of Energy). He published one book on AI applications for the types of NOVA Science, USA.

In his 24 year diverse management experience, ranging from founder and co-owner of a process engineering consulting firm as a consultant for AGIP S.p.A. and AGIP Petroli designed an all energy (electrical and mechanical) system for Agip Petroli's new employee recreational facility at its new headquarters in Rome showcasing the effective deployment and use of solar energy. Led a 2 year, 30 person project to design and construct China's Rural Energy Resources Training Center in Beijing and coached/mentored Chinese counterparts in designing, building and deploying systems to convert renewable energy into usable energy. Produced prototypes and simulation models for training purposes and provided much needed knowledge transfer to the Chinese Government in availability and use of various energy sources.

In recent years he developed an extensive experience in the application and use of expert systems for energy management applications.