Transactions on Information and Communications Technologies vol 2, © 1993 WIT Press, www.witpress.com, ISSN 1743-3517

A customizable tool for the generation of production-based systems

G. Bittencourt & M. Marengoni
Instituto Nacional de Pesquisas Espaciais - INPE/LAC Caixa Postal 515, 12201-970 - São José dos Campos - SP - Brazil

ABSTRACT

A tool is proposed that integrates a modular graphical interface for image processing and an expert system shell generator. The tool provides three knowledge representation formalisms (logic, frames and semantic nets), forward and backward control strategies, and several conflict resolution methods. These features can be combined to construct expert system shells with different levels of complexity. The tool is intended to facilitate the implementation of expert systems in the domain of image processing, and to be used as a teaching and research laboratory in knowledge representation and expert system architecture.

INTRODUCTION

The representation of knowledge in a declarative structure, and the resulting separation of control and knowledge (Schor [26]), characterizes many knowledge-based systems, specially systems based on the production rule paradigm (Davis and King [11]), as in expert systems (e.g. Buchanan and Shortliffe [9]). This separation and the methodology used to build such systems naturally lead to the development of system building tools: knowledge representation languages (e.g. Bobrow and Winograd [4]; Laurent et al. [17]) and expert system shells (e.g. Brownston et al. [7]; Buchanan and Feigenbaum [8]; Van Melle et al. [30]).

A central problem, when specifying or choosing a system building tool, is to determine which formalism is more adequate to represent knowledge about the intended domain. The implementation of a

given formalism usually implies the compromise between expressiveness and efficiency. The analysis of how efficiency is affected, when expressiveness increases, is an active research field (e.g. Brachman and Levesque [5]; Nebel [20]).

Two solutions are usually available if one needs a large choice of knowledge representation formalisms: (i) to have available a large number of system building tools, or (ii) to use a hybrid system (e.g. Brachman et al. [6]; Vilain [31]). The first solution is not practical if the intended application needs special hardware and software interfaces. The second implies the use of a large and usually complex system even when a simpler tool would be sufficient. A third solution is a tool that provides several knowledge representation formalisms, control strategies, and conflict resolution policies, that allows the user to choose one or more of such facilities in order to build a customized expert system shell.

This paper proposes a tool based on this third solution. The tool also provides a standard graphical interface for image processing, the intended application field of the expert systems to be developed using the tool.

Research and development in image processing is one of the main activities of the Brazilian Space Research Institute (INPE). The analysis and interpretation of images from different sources - remote sensing satellites, meteorological radars, aerial photos, etc - involve several specialized activities, presently performed manually or with the help of a variety of computing systems. These activities range from almost automatic tasks to highly complex manipulations involving a lot of specialized knowledge. On the other hand, the graphical interface needed to implement a computational tool to help in these activities is rather standard and can be built modularly.

The tool integrates a modular graphical interface with a shell building tool. In this environment, it is possible to build a customized expert system shell, according to the level of complexity of the intended problem, and then to apply the usual methodology to develop an expert system to solve the problem. As for the graphical interface, it provides graphical primitives, and also allows for the integration of external algorithms needed for special image manipulations. This tool represents the

artificial intelligence part of a larger project of developing an intelligent laboratory for image processing, where research and product development activities could be performed efficiently.

The paper is organized as follows: in Section 2, the general architecture of the tool is presented. In Sections 3 to 7, the modules of the tool are defined. Finally, in Section 8 some conclusions are presented and directions for future developments are discussed.

GENERAL DESCRIPTION

The tool was implemented in Common Lisp (Steele Jr. [28]). The adopted implementation of Common Lisp permits interfacing with the language C and with the X Windows system.

The tool architecture includes two management modules - the tool interface and the shell generator - and three package libraries - the kernel, the knowledge acquisition interface, and the expert system interface. Each package library consists of a set of Lisp function packages that can be used as building blocks to implement an expert system shell. This architecture is presented in figure 1.

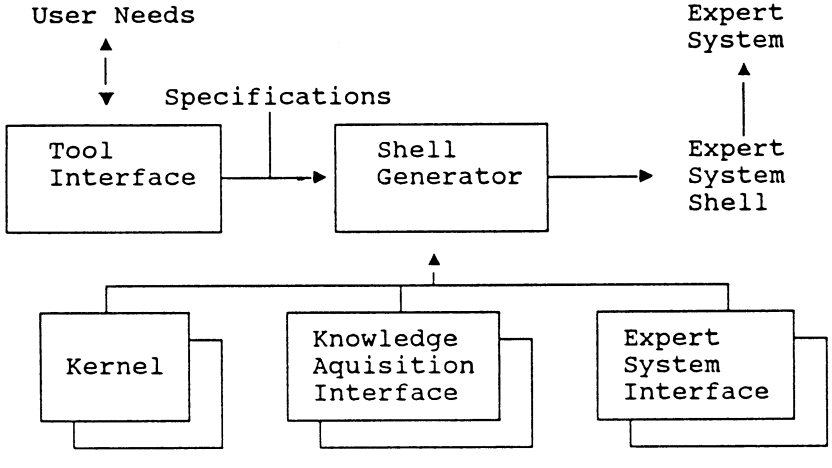

Figure 1. Tool Architecture

The use of the tool is divided in three phases. Initially, the user's needs are specified interactively using the tool interface. In the second phase, these specifications are used by the shell generator to select, from the package libraries, the

ones containing the functions needed to build an expert system shell satisfying the specifications. These packages are integrated in a stand-alone system, which is compiled. Finally, the compiled version of the specified shell is used to develop an expert system to solve the intended problem.

In the following sections, the modules of the tool are described.

TOOL INTERFACE

The tool interface is a menu driven system that allows the user to choose among the facilities provided by the tool. The expert system shells generated by the tool consist of the following modules: knowledge representation, control strategy, conflict resolution, knowledge acquisition interface, and system interface. Each of these modules correspond to a menu entry. To build a customized shell, the user must choose one option for each of these entries. The first three modules are built from packages belonging to the kernel, and should be specified first because the interface options depend on these choices.

When the choice is completed the user may store it in a disk file or proceed to the following phase, calling the shell generator from the appropriate menu option. A further option is available: to execute directly from the tool interface an interpreted version of the specified shell. This option is useful if the user has doubts about his/her choices and wants to perform some preliminary tests.

On line help is provided at all menu levels. Because of the importance of the choice of the adequate knowledge representation formalism, the help associated with this option is the most detailed. Short descriptions of formalisms, their syntaxes and reasoning methods, illustrated with examples, are available. With the aid of examples, the control strategy and the conflict resolution helps explain what kind of problems each formalisms is most adequate to, and how to combine formalisms.

The knowledge acquisition and system interfaces are largely determined by the fixed rule format and the chosen knowledge representation formalisms. Choices are only available on syntactic details and interface layout. Their help options are limited to examples of how these interfaces should be used.

SHELL GENERATOR

The shell generator is a noninteractive program that takes as input an expert system shell specification, generated through the tool interface, and produces a compiled version of a stand-alone system satisfying the specification.

Basically, the stand-alone system is built up by putting together the needed packages. However, some optimization is also necessary, mainly to eliminate unnecessary tests related to options which are not present in the specification.

KERNEL

The kernel consists of the basic packages needed to implement an expert system shell: the rule base manipulation package, the knowledge representation packages, the control strategy packages and the conflict resolution packages.

The integration of different types of packages in coherent expert system shells is achieved through the definition of an uniform interface for the same type of packages, and a coherent interface between different types of packages as will be shown below. These interfaces are specified as formal languages and their format is intended to be efficient rather than user-friendly.

Rule Base

There is only one rule base package, and the formalism implemented in this package determines all the interfaces between packages. This formalism allows for rule bases consisting of independent rule sets. Each rule is represented by a structure of the following form:

```
(Rule
     Name: <string>
     Left-hand side: <pattern>
     Right-hand side: <pattern>
     Next-rules: <list of rule names>
     Previous-rules: <list of rule names>
```

The next and previous attributes are used to define a partial order among the rules. This partial order is necessary to implement some of the conflict resolution methods. Each pattern consists of a conjunction of logical terms, which usually include variables. The result of matching a pattern against a

knowledge base is a list of substitutions, one for each instance of the pattern present in the knowledge base. These substitutions are applied to the pattern in the other side of the rule. How the patterns and the terms are interpreted depends on the adopted knowledge representation formalism and control strategy. The patterns can also include graphical commands, which are procedure calls to the graphical interface provided by the tool.

Knowledge Representation

A knowledge representation package consists of an internal knowledge base, and a set of data manipulation procedures. Internal knowledge bases are divided into three independent parts: facts, goals and hypotheses. How these partitions are interpreted depends on the control strategy. Each element in the internal knowledge base is associated to a time measure. This time measure is based on the fact that expert systems execution can be divided in cycles. The cycle number in which a knowledge element is stored is used as its initial time. This time is used in some conflict resolution methods. The set of manipulation procedures implements a knowledge representation formalism. Three formalisms are available: logic, frames and semantic nets. Each formalism is defined by a formal language, a reasoning mechanism and an interface between the formalism and the rule bases.

The formal language defines the form of the valid expressions that can be stored in the knowledge base of the package. The reasoning mechanism specifies how information not explicitly present in the knowledge base can be inferred, and depends on the specific formalism, some examples being term unification, logical deduction, hierarchical inheritance, default reasoning and procedural inference. The rule base interface defines how a term appearing in a rule pattern is interpreted by the formalism. This usually involves tasks as: matching information already stored in the knowledge base, storing and deleting information and executing an external procedure.

Logic The logical formalism has two main advantages: the inherent expressiveness of language (Hayes [14]), and the well defined and studied semantics (Tarski [29]). It is adequate to domains where the knowledge is largely unstructured and consists of a collection of independent facts. The main drawback of the formalism is the inefficiency of the inference method: automatic deduction.

The tool provides two logical formalisms: propositional logic and first-order logic. The internal knowledge base of these formalisms consists of a list of logical expressions constructed according to the usual logical syntax.

Two reasoning methods are available to each one of the formalisms: unification and theorem proving. The unification method consists of matching a given logical term, appearing in a rule pattern, with the contents of the knowledge base. The logical terms and the knowledge base expressions are always stored in a canonical form to avoid checking for commutativity and associativity of operators. This reasoning method corresponds to a data base access using logical expressions with variables as a query language.

The theorem proving method consists of using a resolution process to prove that the set of knowledge base expressions together with the negation of a term, appearing in a rule pattern, is unsatisfiable. Several control strategies for the resolution process are supplied.

To improve efficiency the list of knowledge base expressions is stored using the Common Lisp facility of hash table. Both methods return a list of substitutions.

The interfacing process between the formalism and the rule base consists of choosing the term to be used in the reasoning process, either from the left-hand side or from the right-hand side pattern of a rule, according to the control strategy. The resulting substitutions are applied to the pattern from the other side and the result is stored in the knowledge base.

Frames The frame formalism was first proposed by Minsky [19]. It consists of a hierarchy of data structures called frames. Each frame is composed of a set of slots to which values can be associated. These values can be any primitive information about the concept represented by the frame, or another frame. Three inference mechanisms are integrated into the formalism: (i) inheritance through the frame hierarchy, (i) default values that can be used when no information is available, and (iii) procedural attachment to allow the execution of external functions.

The formalism is adequate to taxonomically

structured domains where the inheritance mechanism can be efficiently explored. The characteristics of technical knowledge - such as machine descriptions, process descriptions, technical terms, troubleshooting strategies, etc - make the frame formalism a preferential choice when building expert system knowledge bases for those domains (Fikes and Kehler [12]).

The internal knowledge base of the frame formalism consists of a forest of graph-like structures where the nodes are frames. This formalism uses two different formal languages: the knowledge acquisition language and the access language.

The knowledge acquisition language is used to construct an initial frame forest. An empty frame, called TOP, is available to be used as standard top of hierarchy. The value of a slot can be a Lisp object, or a frame name. "Demons", i.e. Lisp functions, can be attached to frame slots and executed when the slot value is accessed, modified or deleted.

The reasoning process consists of verifying whether there are frames in the frame forest that satisfy a term of a rule pattern. The reasoning algorithm searches for inherited slot values, and takes into account the different types of value associated to each slot. This process results in a list of substitutions.

As in the logical formalism, the interface process between the formalism and the rule base consists of choosing which side of the rule to use in the reasoning process, according to the control strategy. The resulting substitutions are applied to the pattern from the other side and the result is used as a conjunction of knowledge acquisition expressions. A knowledge acquisition expression preceeded by the operator NOT is interpreted as a "delete slot value" command.

Semantic Nets The semantic net formalism was introduced by Quillian [24] as a model for human associative memory. It consists of a set of nodes connected by edges. The formalism does not have a generally agreed semantics: the nodes can be used to represent concepts, predicates, objects or whatever, and the edges are associated to arbitrary binary relations between the nodes. Its main inference mechanism are inheritance through the net and network matching. The flexibility of the formalism makes it a

good choice for expert system knowledge representation, but the user must follow some discipline in order not to misuse the formalism capacities.

The internal knowledge base of the semantic net formalism also consists of a forest of graph-like structures, but in this case the nodes are just symbols with associated values and the arcs have labels. Like the frame formalism, the semantic net formalism also uses two different formal languages: the knowledge acquisition language and the access language.

The knowledge acquisition language is used to construct an initial semantic net forest. The value of a node can be any Lisp object. For all semantic net queries appearing in a given expression, the reasoning process consists of verifying whether there is a path of edges, labelled with a given edge name, between two nodes. The positive IS-A query results true if all edges of the path are labelled with positive label names. The negative IS-NOT-A query results true if the exception determination algorithm, based on the *skeptical inheritance strategy* (Horty et al. [16]), returns true. In the simplest case, this corresponds to a path of positively labelled edges, except the last one.

As in the previous formalisms, the interfacing process between the formalism and the rule base consists of chosing the side of the rule to use in the reasoning process, according to the control strategy. If the result is true then the pattern from the other side is used as a knowledge acquisition expression.

Control Strategy

A control strategy package consists of a set of functions to control the general behavior of the generated expert system shells. Its tasks are: to direct the rule applications according to a given chaining strategy (Rychener [25]), and to apply a certitude coeficient mechanism when appropriate (Pearl [23]).

Rules can be applied either in forward or backward directions. In the forward direction, the fact part of the knowledge base is used to match the left-hand side of the rules and the resulting list of substitutions is applied to the right-hand sides, which are interpreted as new facts. This strategy is suitable for data driven problems. In the backward

direction, the hypothesis part of the knowledge base is used to match the right-hand side of the rules and the resulting list of substitutions is applied to the left-hand sides, which are interpreted as new hypotheses. This strategy is used in diagnostic problems.

The independence of facts and hypotheses permits a mixed strategy with the rules being applied in both directions. The implementation of such a mixed strategy is the task of the expert system interface package.

Conflict Resolution

To tackle varied and unexpected demands from the environment, a production system should have two important characteristics: sensitivity and stability (McDermott and Forgy [18]). A system displays sensitivity if it is responsive to changes in the environment and it displays stability if it is able to maintain continuity in its behavior. The function of conflict resolution methods is to provide a mechanism that preserves sensitivity and stability without loosing production autonomy.

The tool provides four classes of conflict resolution methods: (i) production order, (ii) special cases, (iii) recency, and (iv) distinctiveness.

If the production order method is adopted, the user should define a partial or total order among the rules (Georgeff [13]). At execution time, rules with higher priority are chosen to fire.

A special case method is based on some special case relationship between the rules. Two special case methods are available: rule subsumption, and rule instantiation subsumption. In the former, rule terms are compared for inclusion before the application of the matching substitution, and in the latter, after the substitution is applied.

Recency methods use the time measure associated with each element of the knowledge base to decide which rule to fire. Three methods are provided, their difference relying on how the time measure is used to compare rule recency: (i) the rule to be preferred is the one that matches the most recent data element, (ii) the rule that have the most recent element as its least recent term is to be preferred, and (iii) all terms in the rules are compared to each other and an order is determined according to the number of

matched elements with the same time value.

Distinctiveness methods select rules according to the similarity of the rules in the conflict set to the rules already fired. Two distinctiveness methods are provided, one using uninstantiated rules and the other using instantiated rules.

Conflict resolution methods may be combined into conflict resolution strategies, where two or more methods are applied in sequence or in parallel.

KNOWLEDGE ACQUISITION INTERFACE

In order to efficiently provide the integration of different knowledge representation, control strategy and conflict resolution methods, it is necessary to specify a formal interface between the different kernel packages. This interface is not suitable for human interaction, and the goal of the knowledge acquisition interface is to provide a front-end able to translate a user-friendly notation into the internal format used by the kernel.

Knowledge acquisition is considered the main bottleneck in the development of expert systems (Anjewierden [1]), and there is no general theory of how it should be done. The knowledge acquisition interface provided by the proposed tool is based on a natural language tool developed at INPE (Oliveira [21]). This interface translates a subset of a natural language to a formal language determined by the fixed internal rule format and by the formal language associated with the chosen knowledge representation formalisms.

EXPERT SYSTEM INTERFACE

The expert system interface is the final user front-end. It allows the use of an expert system developed with an expert system shell generated by the tool, and supposes that all the rules are already acquired. Its function is to control the problem solution process of the expert system.

This interface is controlled through a menu driven mechanism. Its tasks are: to receive an initial situation description; to ask questions to the user when appropriate; to allow browsing into the rule and knowledge bases; and to explain how the solutions were found. The interaction between user and interface is done with the help of graphic information and through the same natural language

tool used in the acquisition interface.

The interface is connected to the kernel packages by a general control package. This package consists of control functions (to be called by the interface implementation) that manage the behavior of the expert system, according to the kernel representation languages. It consists of a set of auxiliary functions (to access the rule and knowledge base) used in the browsing and explanation tasks, and a main function that performs one processing step. This main function has the following algorithm:

1. Select a rule set to be matched according to the context control policy.

2. For each rule, select the left or right-hand side according to the control strategy, and classify the rule terms according to the knowledge representation formalisms present in the shell.

3. Call the adequate reasoning methods to generate a list of substitutions, one for each instance of the rule pattern present in the appropriate knowledge bases.

4. Execute the conflict resolution method on the set of rule instances.

5. Execute the graphical primitives associated with the chosen rules.

6. Call the appropriate functions to apply the other side rule terms according to the knowledge representation formalisms present in the shell.

7. Update the confidence measure of the modified information in the knowledge bases according to the chosen certitude coeficient mechanism.

APPLICATION

An application developed using an expert system shell generated by the tool, is an expert system to interpret meteorological radar images (Silva [27]). Meteorological phenomena appear on a radar image as cells inside cells, with different intensities. This hierarchical characteristic led to the choice of the frame formalism to represent the knowledge of the domain. The fact that the problem is data-driven imposed the choice of a forward control strategy. According to these decisions, the tool was used to generate an expert system shell with only one

knowledge representation formalism (frames) and only one control strategy (forward). This expert system shell was compiled in a stand-alone system and was used as the basis for the development of the expert system.

The goal of the expert system is to receive successive radar images separated by a time interval, to determine intensity, distance, area, velocity and growing rate of the meteorological phenomena; in addittion, by following specialized interpretation rules, the expert system decides whether the images should be catalogued or not, and whether alerts or alarms should be fired.

The knowledge base is organized as a frame hierarchy. In the first level of the hierarchy, the frames which contain the description of protection areas are defined, and the prototypical frames for image attributes and phenomena evolution data. The functions ("demons") used to calculate phenomena characteristics from the raw data are attached to these frames. Below this fixed, first level, the system constructs a dynamic frame hierarchy containing the information about particular images and their phenomena.

The rule base consists of a set of independent *knowledge sources*, according to the *blackboard* model (Hayes-Roth [15]). Each knowledge source consists of a set of rules designed to tackle a specific problem. The knowledge sources are the following: alert, alarm, cataloging and coordination. The coordination knowledge source controls the global behavior of the system.

The first version of the expert system contains about 50 rules and interprets with success *Plan Position Indicator (PPI)* and *Range-Height Indicator (RHI)* maps, generating the necessary alerts and alarms. The system is presently being extended to interpret also *Constant-Altitude Plan Position Indicator (CAPPI)* maps, and its final version should include about 100 rules.

CONCLUSION

A tool for the generation of expert system shells has been presented. The tool implementation has been motivated by the need to construct expert systems with different levels of complexity in the domain of image processing. Besides the necessary graphical interface, the tool provides several knowledge

representation formalisms, control strategies and conflict resolution methods. A further motivation was the possibility of using such a tool as a teaching and research laboratory in knowledge representation and expert system architecture. The tool has been implemented in Common Lisp and is flexible enough to allow the easy integration of new formalisms, if the necessary interfaces are developed.

Besides the application described above, two other applications are being developed using expert system shells generated with the tool: a remote sensing satellite image interpretation system and a mathematical morphology system for image manipulation.

Planned extensions include the integration of a full hybrid system as a possible knowledge representation formalism. The hybrid system to be adopted, called MANTRA (Bittencourt [3]; Calmet et al. [10]), provides three different knowledge representation formalisms: four-valued logic (Belnap [2]; Patel-Schneider [22]), terminological language, and semantic nets. The system also provides hybrid inference algorithms that allow these formalisms to be used in a combined way. A further possible extension is to include in the tool a rule compilation module. All the basic primitives necessary to implement rule compilation methods are already available in the tool, and these methods can be very useful to improve efficiency in some types of production rules.

ACKNOWLEDGEMENT

This work was partially supported by Fundação de Apoio à Pesquisa do Estado de São Paulo (FAPESP) contract No. 91/3532-2.

REFERENCES

1. Anjewierden, A., 'Knowledge Acquisition Tools.' *AICOM*, Vol. 0, No. 1, pp. 29-38, August 1987.
2. Belnap, N.D., 'A Useful Four-Valued Logic.' In "J.M. Dunn and G. Epstein (Editors), *Modern Uses of Multiple-Valued Logics*", D. Reidel Pub. Co., 1977.
3. Bittencourt, G. *An Architecture for Hybrid Knowledge Representation*. Ph.D. Thesis, Universität Karlsruhe, 31 Januar 1990.
4. Bobrow, D.G. and Winograd, T., 'An Overview of KRL, A Knowledge Representation Language.' *Cognitive Science*, Vol. 1, Num. 1, pp. 3-46, 1977.

5. Brachman, R.J. and Levesque, H.J., 'The Tractability of Subsumption in Frame-Based Description Languages.' *Proceedings of AAAI-84*, pp. 34-37, 1984.
6. Brachman, R.J., Gilbert, V.P. and Levesque, H.J., 'An Essential Hybrid Reasoning System: Knowledge and Symbol Level Accounts of KRYPTON.' *Proceedings of IJCAI 9*, pp. 532-539, 1985.
7. Brownston, L., Farrel, R., Kanr, E. and Martin, N., *Programming Expert Systems in OPS-5, An Introduction to Rule-Based Programming*. Addison Wesley, Reading, MA, 1985.
8. Buchanan, B.G. and Feigenbaum, E.A., 'DENDRAL and Meta-DENDRAL: Their Applications Dimension.' *Artificial Intelligence*, Vol. 11, Num. 1-2, pp. 5-24, August 1978.
9. Buchanan, B.G. and Shortliffe, E.H., *Rule-Based Expert Systems, the MYCIN Experiments of the Stanford Heuristics Programming Project*. Addison Wesley, 1984.
10. Calmet, J., Tjandra, O. and Bittencourt, G. 'MANTRA: A Shell for Hybrid Knowledge Representation.' In: "S. Lee, B. Wah, N.G. Bourbakis and W.T. Tsai (Editors), *Proceedings of the Third International Conference on Tools for Artificial Intelligence*, IEEE Computer Society Press, 1991", San José, California, November 10-13, 1991.
11. Davis, R. and King, J.J., 'An Overview of Productions Systems.' In "E. Elcock and D. Michie (Editors), *Machine Intelligence 8*", Ellis Horwood, Chichester, England, pp. 300-332, 1977.
12. Fikes, R.E. and Kehler, T., 'The Role of Frame-Based Representation in Reasoning.' *Communications of the ACM*, Vol. 28, No. 9, pp. 904-920, September 1985.
13. Georgeff, M.P., 'Procedural Control in Production Systems.' *Artificial Intelligence*, Vol. 18, pp. 175-201, 1982.
14. Hayes, P.J., 'In Defence of Logic.' *Proceedings of IJCAI5*, pp. 559-565, 1977.
15. Hayes-Roth, B., 'A Blackboard Architecture for Control'. *Artificial Intelligence*, Vol. 26, No. 3, pp. 251-, July 1985.
16. Horty, J.F., Thomason, R.H. and Touretzky, D.S., *A Skeptical Theory of Inheritance in Nonmonotonic Semantic Nets*. Technical Report CMU-CS-87-175, Computer Science Department, Carnegie-Mellon University, Pittsburgh, PA, October 1987.
17. Laurent, J.-P., Thome, F., Ayel, J. et Ziebelin, D., 'Kee, Knowledge Craft et Art: Evaluation comparative de trois outils de développement de systèmes experts.' *Revue d'Intelligence*

Artificielle, Vol. 1, No. 2, pp. 25-53, 1987.

18. McDermott, D. and Forgy, C., 'Production System Conflict Resolution Strategies.' In "D. Waterman and F. Hayes-Roth (Editors), *Pattern Directed Inference Systems*", pp. 177-199, Academic Press, New York, 1978.
19. Minsky, M., 'A Framework to Represent Knowledge.' *The Psychology of Computer Vision*, McGraw-Hill, pp. 211-277, 1975.
20. Nebel, B., 'Computational Complexity of Terminological Reasoning in BACK.' *Artificial Intelligence*, Vol. 34, pp. 371-383, 1988.
21. Oliveira, C.A. de, *IDEAL, a Dialogue Interface in Natural Language for Expert Systems (in portuguese: IDEAL, uma interface dialógica em linguagem natural para sistemas especialistas)*. Ph.D. Thesis, Instituto Nacional de Pesquisas Espaciais, São José dos Campos, SP, Brasil, INPE 5151/TDL 424, 1990.
22. Patel-Schneider, P.F., 'A Decidable First-Order Logic for Knowledge Representation.' *Proceedings of IJCAI 9*, pp. 455-458, 1985.
23. Pearl, J., *Probabilistic Reasoning in Intelligent Systems: Network of Plausible Inference*. Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1988.
24. Quillian, M.R., 'Semantic Memory.' In "M.L. Minsky (Editor), *Semantic Information Processing*", pp. 216-270, M.I.T. Press, Cambridge, MA, 1968.
25. Rychener, M.D., 'Control Requirements for the Design of Production System Architectures.' *ACM SIGPLAN/SIGART Newsletter*, pp. 37-44, 1977.
26. Schor, M.I., 'Declarative Knowledge Programming : Better than Procedural ?' *IEEE Expert*, pp. 36-43, Spring 1986.
27. Silva, F. de A.T.F. da, 'A Hybrid Formalism for Representation and Interpretation of Image Knowledge.' *Proceedings of the International Society for Photogrammetry and Remote Sensing Congress*, Washington, DC, August 2-14, 1992.
28. Steele Jr., G.L., *Common LISP, The Language*. Digital Press, Burlington, 1984.
29. Tarski, A., 'The Concept of Truth in Formalized Languages.' In "A. Tarski (Editor), Logic Semantics and Mathematics, Oxford, Clarendon Press, 1956.
30. Van Melle, W., Shortliffe, E.H. and Buchanan, B.G., 'EMYCIN A Domain Independent System that Aids in Constructing Knowledge-Based Consultation Programs.' In:"*State of the Art Report on Machine Intelligence*", New York, Pergamon, Infotech, 1981.
31. Vilain, M., 'The Restricted Language Architecture of a Hybrid Representation System.' *Proceedings of IJCAI 9*, pp. 547-551, 1985.