An Experimental Comparison of Knowledge Representation Schemes

Article *in* Ai Magazine · June 1984

DOI: 10.1609/aimag.v5i2.435 · Source: DBLP

3 authors, including:

Koji Sasaki

1 PUBLICATION **43** CITATIONS

SEE PROFILE

Hirokazu Ihara

57 PUBLICATIONS **1,087** CITATIONS

SEE PROFILE

AI Magazine Volume 5 Number 2 (1984) (© AAAI)

An Experimental Comparison of Knowledge Representation Schemes

Kiyoshi Niwa
Koji Sasaki
Hirokazu Ihara

Systems Development Laboratory
Hitachi, Ltd.
1099 Ohzenji
Asao-ku
Kawasaki, 215, Japan

Abstract

Many techniques for representing knowledge have been proposed, but there have been few reports that compare their application This article presents an experimental comparison of four knowledge representation schemes: a simple production system, a structured production system, a frame system, and a logic system. We built four pilot expert systems to solve the same problem: risk management of a large construction project Observations are made about how the structure of the domain knowledge affects the implementation of expert systems and their run time efficiency

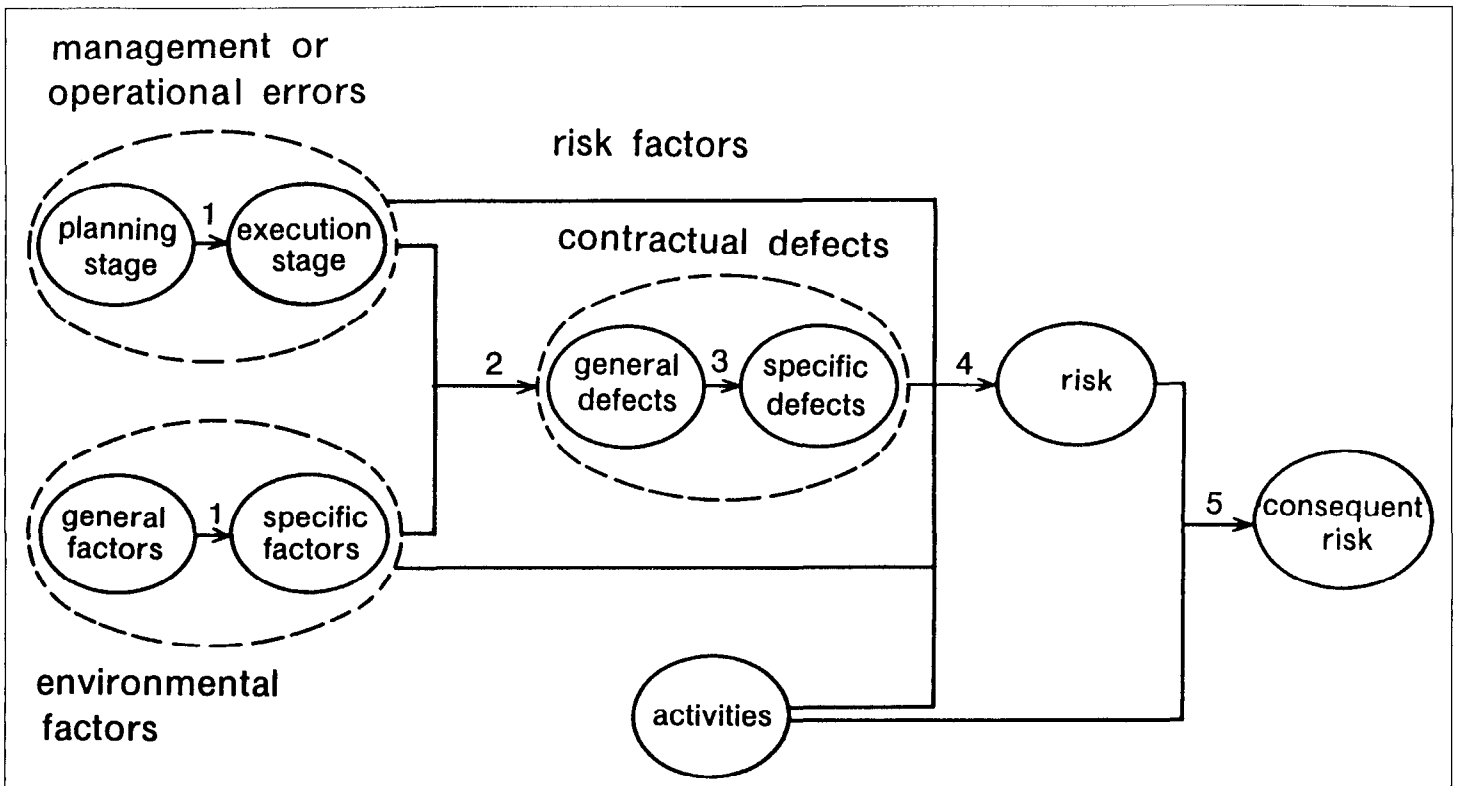
WE THINK THAT IT IS NECESSARY to clarify the advantages and disadvantages of knowledge representation techniques from the expert system designer's point of view. Barr and Feigenbaum (1981) point out that "many researchers feel that the representation of knowledge is the key issue at this point in the development of AI."

Stefik *et al* (1982) point out some domain characteristics that affect expert system design: large search spaces, a need for tentative reasoning, time varying and noisy data. These four characteristics are elaborated into eleven case studies, and some guidelines for expert systems construction are provided. This information helps an expert system designer clarify the domain's characteristics and develop a conceptual system design. However, little information is provided for selecting adequate techniques after the system's function (input/output) is determined.

Ennis (1982) reports her experiences with building an expert system using several tools such as EXPERT, UNITS, EMYCIN, and OPS-5. The system was designed to interpret the X-ray powder diffraction spectra of rocks to determine their constituent minerals. This article focuses on expert system building tools; however, there may be many cases where no such tools are available.

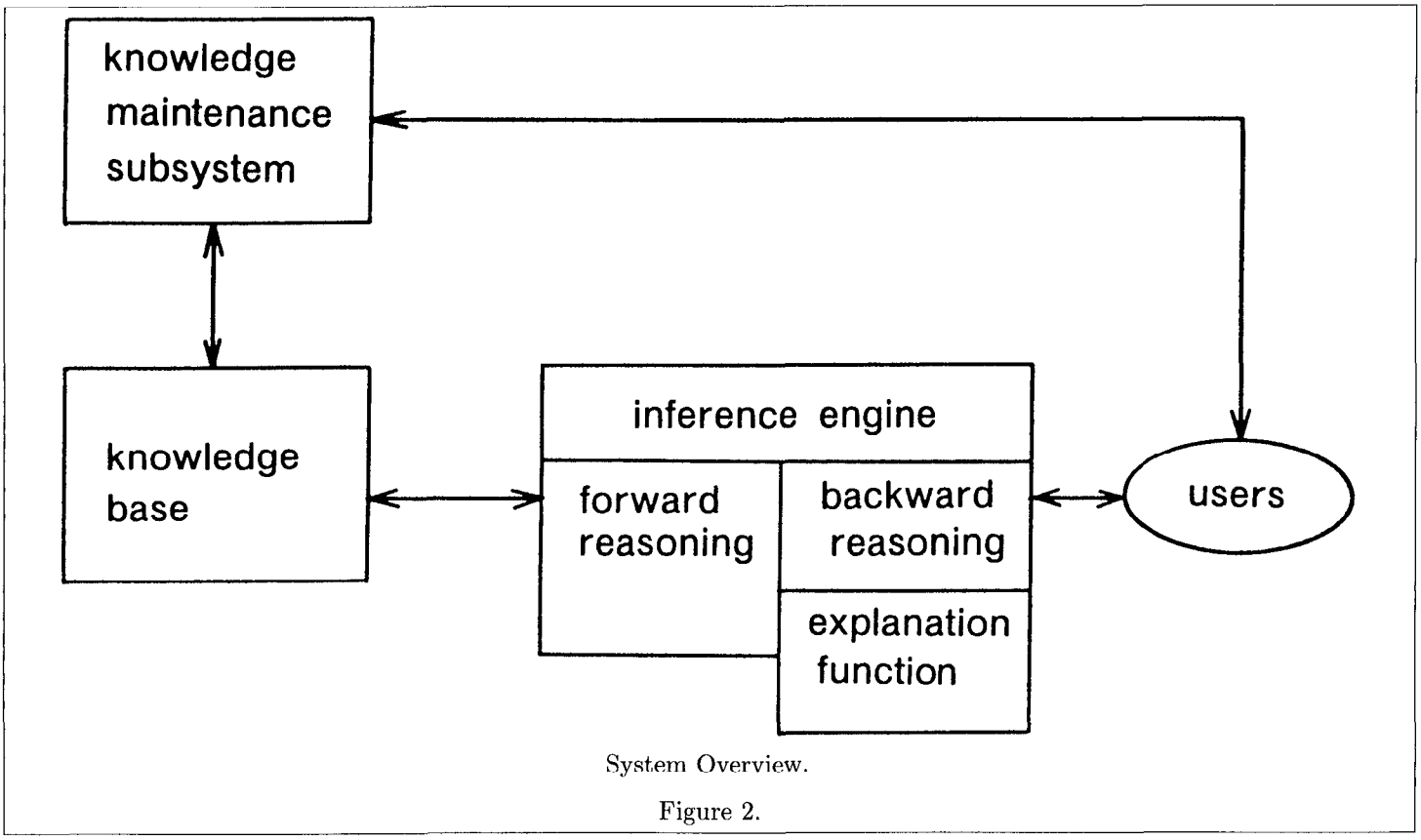
This article's purpose is to present an experimental comparison of four pilot expert systems whose knowledge representations are a simple production system, a structured production system, a frame system, and a logic system. The domain of the four systems is the risk management of large construction projects. Each system's function (inputs/outputs) is exactly the same: to assist a project manager effectively in controlling risks as they arise during project execution. We will compare the difficulties of implementing the knowledge base and the inference engine, as well as run time efficiency. The systems were implemented on a VAX 11/780 using Franz-LISP.

The authors would like to express their appreciation to Dr Edward A. Feigenbaum and H Penny Nii of Stanford University for discussing the early results of this research during their visit to the authors' laboratory The authors also would like to acknowledge the support of Dr Jun Kawasaki, the general manager of Systems Development Laboratory, Hitachi, Ltd



Domain Knowledge Structure Model.

Figure 1.



System Overview.

Figure 2.

Application Description

The Domain

The problem used for this study is risk management of large construction projects. Risks are defined here as undesirable events causing project delays, cost overruns or deficiencies in technical performance. This problem was chosen because Niwa had previously developed a system for this particular domain (Niwa and Okuma, 1982; Niwa and Sasaki, 1983).

The model of domain knowledge relationships is shown in Figure 1. Risk causes have been classified into three groups: management or operational errors, environmental factors, and contractual defects. These factors, errors and defects interrelate with one another to cause risks in certain activities. The model presents a risk-to-risk consequent relationship, *i.e.*, if no countermeasure is taken for a risk, further undesirable events may occur. The relationships in the model flow in one direction (from left to right) along a time dimension.

Characteristics Common to All Four Systems

All four systems were developed on the VAX 11/780 using Franz-Lisp. Each system consists of a knowledge base, an inference engine, and a knowledge maintenance subsystem. The inference engine is capable of both forward and backward reasoning, with the latter including an explanation function. The knowledge maintenance function helps a user add, replace or delete knowledge in the knowledge base. The system model is shown in Figure 2.

Our objective is to help project managers effectively control their projects by providing them with appropriate knowledge gathered from many expert project managers and fused into the knowledge base. In forward reasoning mode, the systems are designed to warn the user (project manager) of risks that could follow from causes entered by the user. In backward reasoning mode, the user enters a hypothetical risk, which the system confirms or denies as possible based on its model of risk causes.

Forward Reasoning. These systems use forward reasoning to inform the user of consequent risks that could follow the specific risk causes.

The user chooses one of two ways for inputting risk causes: the menu method or the key word in context [KWIC] method The menu shows all causes to the user who selects the appropriate ones. In KWIC, the user inputs key words or phrases; then risk causes containing these words or phrases are shown. The latter case, represented in Figure 3, shows a case in which the user has input three key phrases: 'customer', 'law', and 'project manager'. Among risk causes involving at least one of the three strings, he selects 3K01 (complicated or foreign laws), 3Q03 (different business practices of customers) and 2G31 (lack of examination by project managers). Next, he inputs codes for the activities that he wants the system to analyze for possible risks.

```
please key-in strings for risk causes or?
)customer, law, project manager
*please select risk cause codes
1g05   contractual defect in time for approval
3O01   lack in customer english ability
3Q01   lack in customer or consultant ability
3Q03   different business practices of customers
3k01   complicated laws or those different from Japan's
3k02   law or regulation change
2e31   project manager misguidance
2g31   lack in examination of project managers
)3k01, 3Q03, 2g31
*please select activity codes or all
)all
rule 1016   deduced(1d02 contractual defect in technical
            guarantee)
            deduced(1f01 contractual defect in material
            standard)
rule 3018   deduced(6105011 spare parts air cargo due to
            incomplete delivery)
***risk alarm***
2103002   approval delay due to misguiding spare parts
          amount
2103011   civil approval delay due to loading data
          between civil and equipment differences
2202013   Additional equipment because no one
          examines all specifications thoroughly
2202012   spare parts re-order due to number misorder
5103013   pipe foundation change for big equipment
          carry-in
2304003   material re-test due to inspection company
          poor interpretation
6105011   spare parts air cargo due to incomplete
          delivery
```

Figure 3. Forward Reasoning Example.

The system first shows the inference process by printing out rules (or frames or clauses) in order, then gives risk alarms. No conflict resolution is performed. In the management domain, it is desirable that all alternatives be shown to the user for evaluation and selection, because the domain is too complex to rely entirely on a fully autonomous decision system.

Backward Reasoning. Backward reasoning was developed for a project manager's "dynamic checklist". When a user inputs a risk as a hypothesis, the system asks him about various conditions one after another until the hypothesis (risk) is determined to be likely. If asked, the system can explain its reasoning process.

These hypotheses also can be entered by the KWIC method. The authors observed that in this domain the users were comfortable with hypotheses expressed as character strings including words, phrases, and sentences

An example in which a user worries about risks relating to a consultant is shown in Figure 4. After the user inputs

```
*please key-in strings for risk
)consultant
*please select risk codes
2103003   consultant's approval delay
41 030    civil work delay due to bad negotiation with
          consultant
51 051    sudden material change due to consultant error
)2103003
*hypothesis 2103003
is this true (2h31 lack in project manager coordination)
)no
is this true 3k01 complicated laws or those different from
Japan's
)no
is this true (3Q01 lack in customer and consultant ability)
)yes
is this true (2e31 project manager misguidance)
)yes
is this true (2103 approval activity)
)yes
*hypothesis 2103003(consultant's approval delay)may oc-
cur. Do you want to know how it is deduced?
)yes
**following rules were used***
rule 3002   (if part 2103007 2103)deduced 2103003
*2103007    material upgrade request for customer's
            future plan
*2103       approval activity(yes input)
rule 2002   (if part 1a01 2103)deduced 2103007
*1a01       contractual defect in scope of equipment
            supply
*2103       approval activity(yes input)
rule 1003   (if part 3Q01 2e31)deduced 1a01
*3Q01       lack in customer and consultant ability
            (yes input)
*2e31       project manager misguidance
            (yes input)
```

Figure 4. Backward Reasoning Example.

'consultant', he is informed of the risks in the knowledge base that include that string. He selects risk 2103003 (consultant's approval delay) as a hypothesis for backward reasoning. The system, after requiring the user's replies (yes or no) for the hypothesis' condition parts, concludes that the hypothesized risk may occur. As an explanation, the system gives rules (or frames or clauses) in their order of inference

Development of the Pilot Systems

Four kinds of pilot expert systems were developed using the same requirements and the same knowledge as described before. These were a simple production system, a structured production system, a frame system, and a logic system.

Simple Production System. As one dimension of causal knowledge is used, it is easily described by the types of production rules shown in Figure 5. Rule 1001 is an example of a rule that deduces risk cause types from risk causes. Rule 2062 is an example of a rule that deduces risk types from risk causes. Rule 3018 is an example of a rule that deduces consequent risk types from risks

Every clause of the production rules is actually represented by four- or seven-character codes for improving matching efficiency. The ASSOC function in LISP combines the code with a translation when the latter is requested. Rule 1001, for example, is stored in the production memory as shown below.

```
(RULE 1001
  (IF (3K01)(2F16))
  (THEN(1A01)(1K01)))
```

A core of forward and backward reasoning algorithms in the inference engine was implemented by applying methods described by Winston and Horn (1981).

Structured Production System. The production rules in the simple production system are divided into five knowledge sources, according to the temporal order in the model of domain knowledge relationships. (The numbers in Figure 1 correspond to the knowledge sources) Knowledge source control functions are added to the simple production system's inference engine.

Frame System. The frame system is implemented by using the method of Winston and Horn (1981), based on the frame representation language [FRL]. Three kinds of frames were made: risk cause, risk, and activity.

Risk frame 2103003 (consultant's approval delay) is shown in Figure 6. It is described as a kind of (AKO) risk

Rule 1001

If (complicated laws or those different from Japan's) and (sales department poor countermeasure)

Then (contractual defect in scope of equipment supply) and (contractual defect in arbitration or force major)

Rule 2062

If (contractual defect in scope of equipment supply) and (delivery)

Then (spare parts amount misunderstand to be delivered) and (special tools excessive request due to incomplete confirmation) and (excessive delivery due to no revision of delivery scope after design change)

Rule 3018

If (spare parts amount misunderstand to be delivered) and (delivery)

Then (spare parts air cargo due to incomplete delivery) and (customer's reject of project acceptance)

Figure 5. Production Rule Example.

2103003	
AKO	risks at hard design approval stage
Name	consultant's approval delay
Risk causes	lack in project manager coordination or investigation (2H31), and lack in customer or consultant ability (3Q01)
Consequent risks	
Consequent risk causes	material upgrade request for customer's future plan (2103007)

Figure 6. Frame Example.

occurring at the hard design approval stage. The risk causes are 2H31 (lack of project manager coordination or investigation) and 3Q01 (lack of customer or consultant ability). Its consequent risk is not recorded thus far. However, the frame shows that risk 2103003 is a consequent risk of risk 2103007 (material upgrade request for customer's future plan), meaning that if no countermeasure is taken for risk 2103007, then risk 2103003 may occur.

The property frame for risk 2103003 is stored as:

```
(2103003
  (AKO(VALUE(HARD-APPROVAL-RISK)))
  (NAME(VALUE(CONSULTANT'S APPROVAL DELAY)))
  (FACTOR(VALUE(2H31 3Q01)))
  (CONSEQUENT-RISK-FACTOR(VALUE(2103007))))
```

Slots in the frame are used to represent risk causality as described above.

AKO inheritance is used in the pilot system; however, procedural attachment is not. The most important function of the frame system's inference engine is to organize basic frame-handling functions so that forward and backward reasoning are performed.

Logic System. Although the propositional logic is sufficient to meet the pilot system requirements, the resolution principle of first order was applied in consideration of future system extension. Chang and Lee's (1973) algorithm is used for the resolution principle program's core. Knowledge is represented in Horn clause form as shown below, because we planned to use PROLOG in the next phase. Production rule 1001, for example, is changed into two Horn clauses:

```
(1521 NIL ((NOT 3K01)(NOT 2F16)(1A01))), AND
(1522 NIL ((NOT 3K01)(NOT 2F16)(1K01)))
```

Control algorithms for forward and backward reasoning were developed by analogy with the production system's algorithms.

Experimental Comparison

Implementation Difficulties

Knowledge base. The volumes of the knowledge bases for the four pilot systems are:

• simple production system	263 rules	15k characters
• structured production system	263 rules	15k characters
• frame system	213 frames	29k characters
• logic system	348 clauses	17k characters

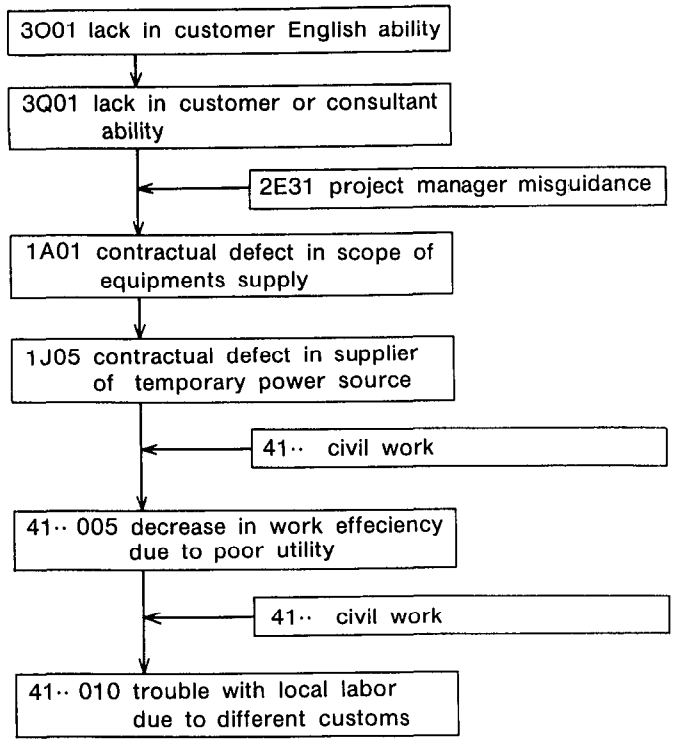
We observed the following:

- The number of rules and characters for the simple and structured production systems is the same, because the simple production system's rules were divided into the five knowledge sources of the structured production system The 213 frames are fewer than the 263 production rules, because some related rules were merged into a single frame.
- The number of characters in the frame system is greater, because it was necessary to replicate some related pieces of knowledge into different frames.
- The number of clauses in the logic system is greater than the number of production rules, because the Horn clause representation was applied.

Our evaluation of the difficulty associated with the implementation of the knowledge bases is based upon our subjective judgment rather than using the number of person-hours spent on the task as an objective measure. We found as we became more experienced, each pilot system was developed in less time and with more facility than the previous one. Our results (in order of difficulty) are as follows:

- The simple production and logic systems' knowledge bases are the most easily implemented because both of these representations are very modular and clearly capture the causal relationships
- The structured production system knowledge base was implemented with some difficulty, because it was necessary to consider how many knowledge sources were adequate and in which knowledge source each rule should be placed
- The frame system was the most difficult to implement because it contains more structure than the other representations. It was necessary to determine what kinds of frames the system needs, what kinds of slots each frame needs, and how all the frames fit together into an AKO hierarchy.

The coding quantity difference between the production and logic systems is not mentioned, because we were easily able to produce the necessary additional coding by changing production rules with plural right-hand side elements into plural clauses.



Inference Chain Example.

Figure 7.

Inference engine. The volumes of the inference engines for the four pilot systems are:

- simple production system 9.1K characters
- structured production system 9.5K characters
- frame system 14 3K characters
- logic system 11.3K characters

Although the basic algorithms for forward and backward reasoning in the production system and the basic unification algorithm for the logic system were taken from published sources, some development was necessary to adapt these to the needs of our pilot systems. In our experience:

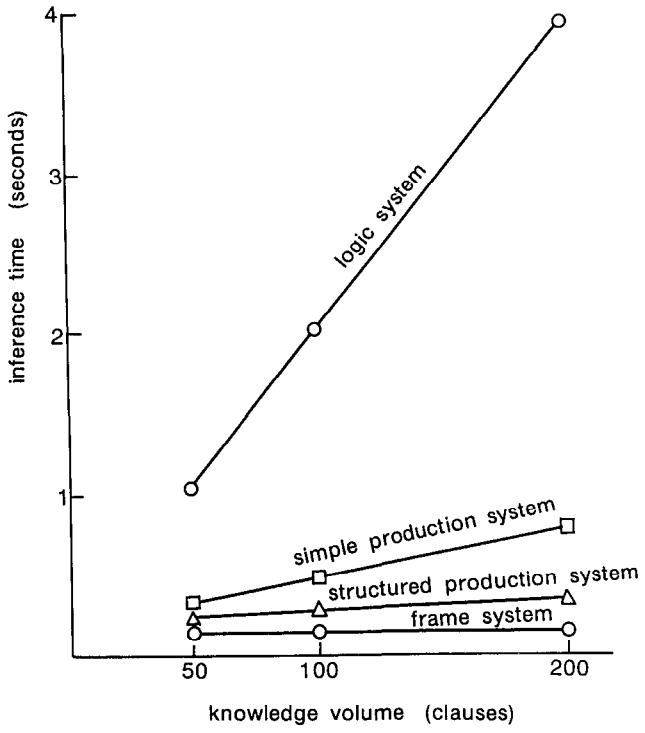
- The simple production system was the easiest to implement, because little adaptation of Winston and Horn's algorithms was necessary
- The structured production system was more difficult, because an algorithm which would control knowledge sources had to be developed.
- The level of difficulty in implementing the logic system was similar to that of the structured production system In addition to the basic unification algorithm, the inference engine needed algorithms for linear resolution and for forward and backward reasoning. An additional difficulty was the necessity of implementing a logically complete system
- The frame system was the most difficult of all to implement, because it was necessary to develop forward and backward reasoning processes on top of Winston and Horn's basic frame-handling functions.

Inference run time efficiency. We measured the CPU times for each system while running the same problem. The test contained both forward and backward reasoning executed along three kinds of inference chains, one of which is shown in Figure 7. Forward reasoning started with 3001 (lack of customer's English ability), 2E31 (project manager misguidance) and 41.. (civil work) as conditions or risk causes. Backward reasoning started from the chain's bottom as a hypothesis.

The size of the knowledge base was varied as an experimental parameter. The number of Horn clauses in the logic system was taken as a standard of comparison. The sizes of the other knowledge bases were adjusted to represent the same knowledge content. The relative size of each knowledge base is shown below:

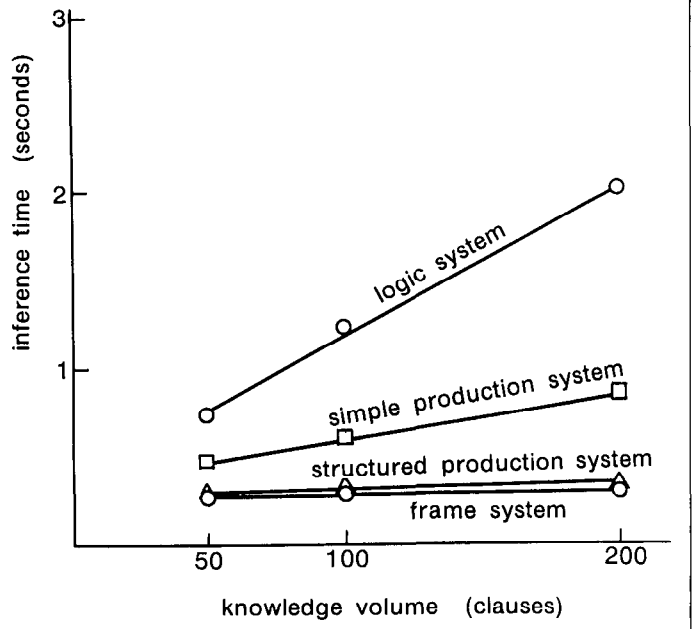
• Number of Horn clauses	50	100	200
• Number of production rules	41	79	157
• Number of frames	96	131	175

The CPU time was measured by the Franz-LISP function PTIME (process time minus garbage-collection time). Measured inference time did not include the user's input



Inference Time and Knowledge Base Volume for Forward Reasoning.

Figure 8.



Inference Time and Knowledge Base Volume for Backward Reasoning.

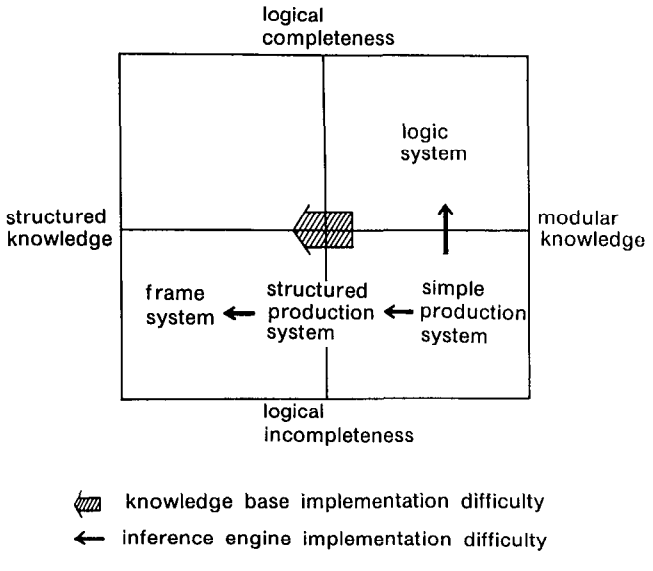
Figure 9.

or the program's output, but did include an explanation of the backward reasoning process. Seventy-two points were measured: 4 pilot systems × 3 problems × 2 types of reasoning × 3 sizes of knowledge bases.

Experimental results. Experimental results for forward and backward reasoning are shown in Figures 8 and 9, respectively. The measured points in the figures are the averages of three kinds of problems. These figures show that for all volumes of knowledge base, the frame system used the least inference time while the logic system used the most. As the amount of knowledge increased, the inference times of the frame and structured production systems remained roughly constant. The inference time of the simple production system increased moderately, while that of the logic system increased markedly.

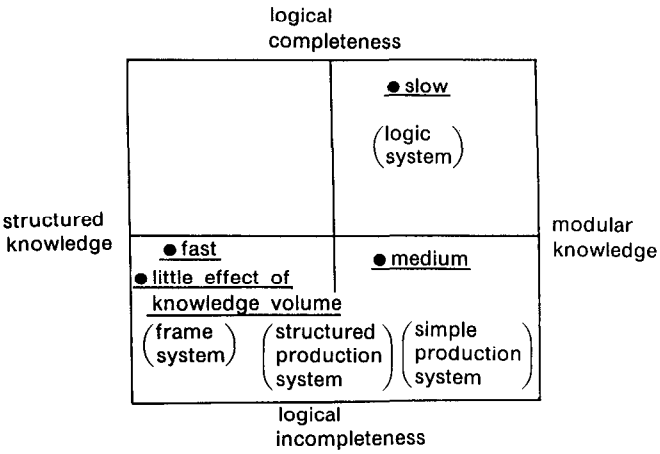
These results are due to the following factors:

- In the frame system, related pieces of knowledge are connected to one another by pointers, thereby limiting search. This means that inference time is low and relatively insensitive to the size of the knowledge base
- The number of rules to be searched in the structured production system is limited compared to that of the simple production system, which again means that inference time is low and not strongly dependent on the size of the knowledge base.
- In the simple production and logic systems, the effects of increasing knowledge volume were significant, because all knowledge had to be searched.
- Resolution in the logic system was relatively time-consuming.



Implementation Difficulties.

Figure 10.



Run Time Efficiency.

Figure 11.

Conclusions

All of the foregoing points are summarized in Figures 10 and 11. The horizontal axis represents knowledge modularity and the vertical axis represents logical completeness. The logic system is modular and logically complete. Other systems are relatively incomplete logically; their knowledge is structured to different degrees.

In Figure 10, the increasing difficulty of knowledge base implementation is shown by the thick arrow; the increasing difficulty of inference engine implementation is shown by the thin arrows. The figure leads to the conclusion that the more structured the knowledge base, the more difficult it is to implement the knowledge base and the inference engine.

Also, it was more difficult to implement the logic system inference engine than that of the simple production system

Figure 11 shows that when knowledge is structured, run time efficiency increases, while sensitivity to the volume of the knowledge base decreases Also, inference in the logic system is slow.

In summary, we can rearrange our findings to make the following statements:

- In a poorly understood domain whose knowledge structure cannot be well described, modular knowledge representations, *e.g* , simple production and logic systems, should be used However, this causes low run time efficiency
- The use of structured knowledge representations, *e.g* , structured production and frame systems, increases run time efficiency as well as reducing the effect of the knowledge volume on run time However, system implementation is more difficult.
- Mathematical completeness makes logic systems more difficult to implement and less efficient in run time Our problem was too simple to adequately demonstrate the advantages of logic representation

There are many reasons for structuring knowledge, of which the most common is effective use. To achieve this end, meta-knowledge for structuring is necessary during the system design phase. This is why the implementation of the knowledge base and inference engine for structured knowledge is difficult. On the other hand, once a structured system is implemented, run time efficiency increases In general, the ease of structuring knowledge in the system design phase depends on the knowledge engineer's ability and the characteristics of the domain. Although the rule compilation technique of Forgy and McDermott (1977) and the cognitive economy proposal by Lenat *et al.* (1979) are approaches to this relatively unexplored problem, further research is necessary.

References

Barr, A and Feigenbaum, E.A. (1981) *The handbook of artificial intelligence* Volume 1 Los Altos: William Kaufmann, Inc.

Chang, C and Lee, R.C. (1973) *Symbolic logic and mechanical theorem proving* New York: Academic Press.

Ennis, S.P. (1982) Expert systems, a user's perspective to some current tools *AAAI-82*, 319 - 321.

Feigenbaum, E A and McCorduck, P (1983) *The fifth generation: artificial intelligence and Japan's computer challenge to the world.* Reading: Addison-Wesley Publishing Company.

Forgy, C A (1979) On the efficient implementation of production systems. Doctoral Diss., Dept of Computer Science, Carnegie-Mellon University

Lenat, D B , Hayes-Roth, F and Klahr, P. (1979) Cognitive Economy. Tech Rep. HPP-79-15, Computer Science Dept , Stanford University.

Niwa, K. and Okuma, M. (1982) Know-How transfer method and its application to risk management for large construction projects *IEEE Transactions on Engineering Management* EM-29 (4): 146 - 153

Niwa, K. and Sasaki, K (1983) A new project management system approach: the "Know-How" based project management system. *Project Management Quarterly* 14(1), 65 - 72.

Stefik, M , Aikins, J., Balzer, R , Benoit, J , Birnbaum, L , Hayes-Roth, F. and Sacerdoti, E (1982) The organization of expert systems: a tutorial. *Artificial Intelligence* 18, 135 - 173

Winston, P.H. and Horn, B K P. (1981) *LISP* Reading: Addison-Wesley.