KBS

WELCOME TO

AN EXPERIMENTAL COMPARISON OF KNOWLEDGE REPRESENTATION SCHEMES

Which AI method works best for expert systems in risk management?

Author : Kiyoshi Niwa, Koji Sasaki, Hirokazu Ihara

supervised by : eng wegdan

Date: April , 2025

Outline

- introduction
- Knowledge Representation Methods
- System Functionality
- Experimental Setup
- Key Findings
- Conclusions & Recommendations
- Q&A / Discussion

introduction

Objective:

Compare 4 AI knowledge representation methods for expert systems.

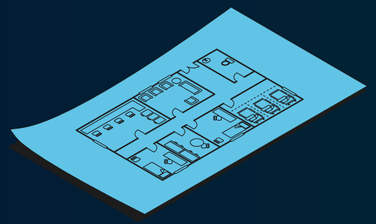
Domain:

Risk management in large construction projects (delays, cost overruns, technical failures).

Why It Matters:

- Poor knowledge representation → Slow, unscalable, or inaccurate Or Unreliable decisions. (trads off)
- Helps developers choose the right balance of speed, scalability, and ease of use.






Knowledge Representation Methods

Simple Production System

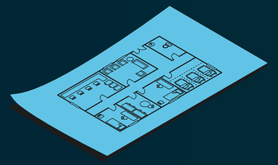
- IF-THEN rules ("If laws are complex → contract defects").

Simple Production System

- Rules grouped into knowledge sources (e.g., "causes," "risks").

Frame System

- Hierarchical objects with slots (e.g., "Risk: Delay" → causes, consequences).



Logic System

- Horn clauses (e.g., "¬complex_laws ∨ ¬poor_planning ∨ defect").

System Functionality

```
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.

Forward Reasoning:

- Input causes → Output risks.

Backward Reasoning:

- Hypothetical risk → Root causes.

User Input → Inference Engine → Risk Alerts.

Like a doctor diagnosing a disease
systems reason forward (symptoms → illness)
or backward (illness → causes

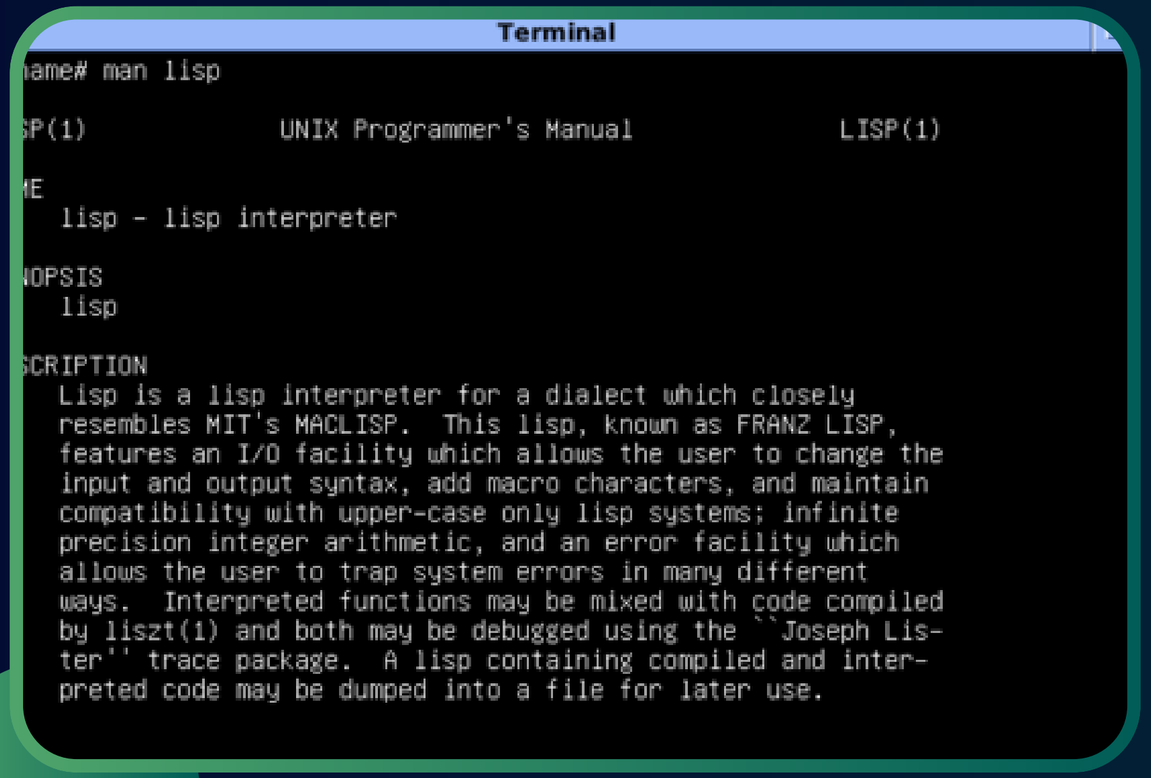
```
please key-in strings for risk
)consultant
*please select risk codes
2103003   consultant's approval delay
41  030   civil work delay due to bad negotiation with
          consultant
51  051   sudden material change due to consultant error
)2103003
*hypothesis 2103003
is this true (2h31 lack in project 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)
```

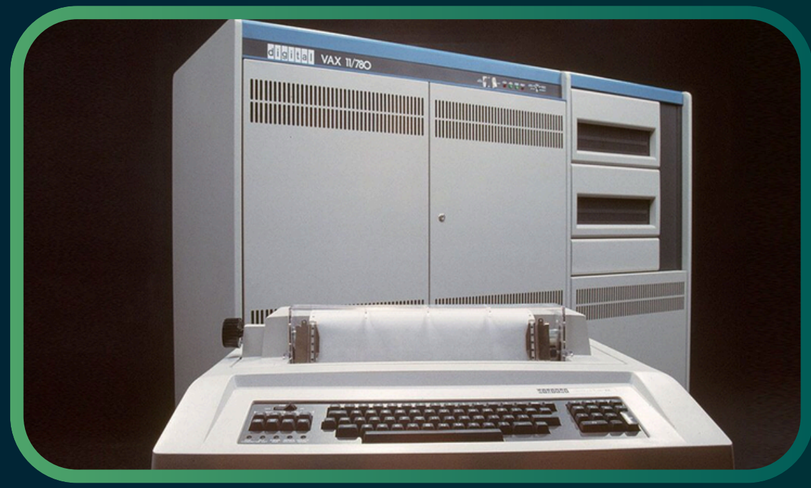
Figure 4. Backward Reasoning Example.

Experimental Setup

Hardware/Software

Franz Lisp programming language





VAX 1980 computer version

Metrics Tested:

- Implementation difficulty (knowledge base, inference engine).
- Runtime efficiency (CPU time).
- Scalability (performance as knowledge grows).

Key Findings

Implementation Difficulty

Easiest to Build:

- **Simple Production Rules** (like basic IF-THEN statements).
- **Logic Systems** (because they use formal math, but only if you're comfortable with logic).

Hardest to Build:

- **Frame Systems** (because you must design complex hierarchies, like folders within folders).

Runtime Efficiency (Speed)

Fastest:

- **Frame Systems** (they use pointers, like shortcuts, to find answers quickly).
- Frames group related knowledge together, so the AI doesn't waste time searching.

Slowest :

- **Logic Systems** (they solve problems step-by-step like a math proof, which takes time).

Scalability (Handling Large Systems)

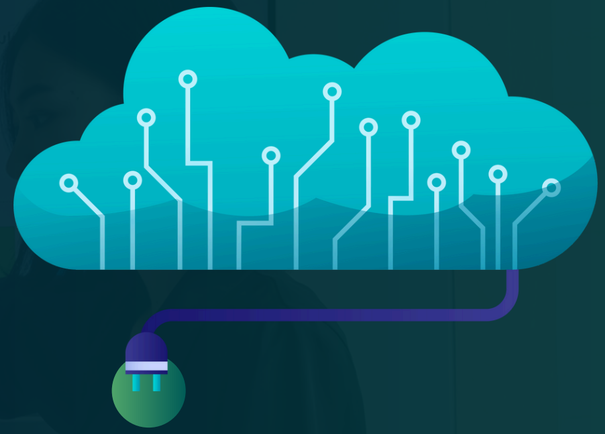
Best for Big Systems:

- **Frame Systems** and **Structured Production Rules** (they stay organized as they grow).

Worst for Big Systems:

- **Simple Production Rules** (they become a tangled mess, like a closet with no shelves).
- **Logic Systems** (they get slower as you add more rules).

Q&A / Discussion



Thank You
any question?

See You Next