List-scan

(defun scan-row
  (board row function)
  (dotimes (column 3)
    (funcall function
      (aref board row column row column)))
(defun list-scan
  (scan-fn board arg)
  (let ((result nil))
    (funcall scan-fn board arg #'
      (lambda (mark row column)
         (push mark result)))
    result))
List-scan

(defun scan-row
  (board row function)
  (dotimes (column 3)
    (funcall function
      (aref board row column
        row
        column)))
(defun list-scan
  (scan-fn board arg)
  (let ((result nil))
    (funcall scan-fn board arg
      #'(lambda (mark row column
            (push mark result)))
      result))
  result)
List-scan

```lisp
(defun scan-row
  (board row function)
  (dotimes (column 3)
    (funcall function (aref board row column row column)))
)

(defun list-scan
  (scan-fn board arg)
  (let ((result nil))
    (funcall scan-fn board arg #'(lambda (mark row column)
                                (push mark result)))
    result))
```

Knowledge-based Agent Architecture

- **Inference Engine**
- **Sensor**
- **Executive**
- **Knowledge Base**
- **Effectors**

Stores true facts

Derives new facts from old
Review

Assertions and Queries
• Executive asserts (tells KB) “I sense ....”
• Executive queries (asks KB) “What should I do?”
• Internal assertions and queries of KB
  – Sense and Do
  – State of the world

Goal:
• System designer enters a set of facts
• System decides for itself how & when to use the facts

Review

• A logic consists of
  – Formal language: sentences
  – Inference mechanism:
• Model: a mapping to the real world
  – Tautology, satisfiable, contradiction
• Validity, soundness, completeness
• Propositional Logic
  – Language: symbols, ¬, ^, v, =>, ( )
  – Inference: truth tables
  – Inference: rules
  – Resolution
Resolution

\[
\alpha \lor \beta \quad \neg \alpha \lor \gamma \\
\therefore \beta \lor \gamma
\]

There is a pit in (1, 2) or there is a pit in (2, 1).
\[
\alpha \lor \beta \\
\therefore \neg \beta \lor \gamma
\]

There is not a pit in (1, 2) or there is a breeze in (1, 3)
\[
\neg \alpha \lor \gamma \\
\therefore \beta \lor \gamma
\]

there is a pit in (2, 1) or there is a breeze in (1, 3)

Example Proof

Meanings:

P12 = There is a pit in (1, 2)
P21 = There is a pit in (2, 1)
B11 = There is a breeze in (1, 1)
B13 = There is a breeze in (1, 3)

Facts:

\[
\neg B11 \lor P12 \lor P21 \quad \text{breeze in (1, 1) } \Rightarrow \text{ pit in (1,2) or (2,1)} \\
\neg P12 \lor B13 \quad \text{pit in (1, 2) } \Rightarrow \text{ breeze in (1, 3)} \\
B11 \quad \text{breeze in (1, 1)} \\
\neg B13 \quad \text{no breeze in (1, 3)}
\]
Proof

\[ \neg B_{11} \lor P_{12} \lor P_{21} \quad B_{11} \]
\[ P_{12} \lor P_{21} \quad \neg P_{12} \lor B_{13} \]
\[ P_{21} \lor B_{13} \quad \neg B_{13} \]
\[ P_{21} \]

Limitations of the Propositional Agent

- Cannot generalize rules over the domain. For example,
  - we need separate rules for each square.
  - Need time-dependent versions of each rule at each square to account for changes in time.
- First-order logic will solve these problems
First-Order Logic

makes stronger ontological commitments than propositional logic.

- the world consists of objects that have properties
- relations, including functions, are defined on
- objects.

Examples

- objects: dogs, computers, people, exams, LISP assignments
- relations: larger than, above, part of, friend of, afraid of
- Properties: green, large, incomplete
- Functions: mother of, one more than

Example Sentences

Brother(Richard, John)
Brother(Richard, John) \land Brother(John, Richard)
Older(John, 30) \implies \neg Younger(John, 30)

- Relations are defined by a set of tuples that satisfy the relation. For
- example, the Brother relation could be defined as \{<John, Richard>, <Richard, John>\}
Quantifiers

Universal Quantification:
- $\forall x \; \text{Cat}(x) \Rightarrow \text{Mammal}(x)$
- means "for any object x, if x is a cat, then x is a mammal."

Existential Quantification:
- $\exists x \; \text{Sister}(x, \text{Morris}) \land \text{Cat}(x)$
- means "Morris has a sister who is a cat"

Nested Quantifiers

- What do the following mean?
  - $\forall x, y \; \text{Parent}(x, y) \Rightarrow \text{Child}(y, x)$
  - $\forall x, y \; \text{Brother}(x, y) \Rightarrow \text{Sibling}(y, x)$
  - $\forall x \; \exists y \; \text{Loves}(x, y)$
  - $\exists y \; \forall x \; \text{Loves}(x, y)$
Connections between Quantifiers

\[ \forall x \neg \text{Likes}(x, \text{Morris}) \equiv \neg \exists x \text{Likes}(x, \text{Morris}) \]
\[ \exists x \neg \text{Likes}(x, \text{Morris}) \equiv \neg \forall x \text{Likes}(x, \text{Morris}) \]

English into first-order logic

– ```You can fool some of the people all of the time.''
– ```You can fool all of the people some of the time.''
– ```You cannot ever fool your mother.''

• **Answers**
  
  \[ (\forall t) (\exists p) \text{Time}(t) ^\text{Person}(p) \Rightarrow \text{Foolable}(p,t) \]
  \[ (\exists p) (\forall t) \text{Time}(t) ^\text{Person}(p) \Rightarrow \text{Foolable}(p,t) \]
  \[ (\exists t) (\forall p) \text{Time}(t) ^\text{Person}(p) \Rightarrow \text{Foolable}(p,t) \]
  \[ (\forall p)(\exists t) \text{Time}(t) ^\text{Person}(p) \Rightarrow \text{Foolable}(p,t) \]
  \[ (\forall t) \neg \text{Foolable}((\text{Mother(you)},t) \]
  \[ (\forall p) (\forall t) \neg \text{Foolable}((\text{Mother(you)},t) \]
First-order logic into English

\( \forall x \) Hesitates\( (x) \) \( \Rightarrow \) Lost\( (x) \)
\( \neg (\exists x) \) Business\( (x) \) \( ^{\land} \) Like\( (x,\text{Showbusiness}) \)
\( \forall x \) Glitters\( (x) \) \( \Rightarrow \) \( \neg \) Gold\( (x) \)
\( \exists x \) Glitters\( (x) \) \( ^{\land} \) \( \neg \) Gold\( (x) \)

- answers
  - He who hesitates is lost.
  - There's no business like show business.
  - All that glitters is not gold.
  - Nothing that glitters is gold.
  - Not all that glitters is gold.

Some Extensions

- Higher-Order Logics allow quantification over functions and relations in addition to objects.
- The lambda-operator builds predicates and functions from simpler components.
- The uniqueness quantifier \( \exists! \) means ``there is exactly one''.