CS 520: Introduction to Artificial Intelligence

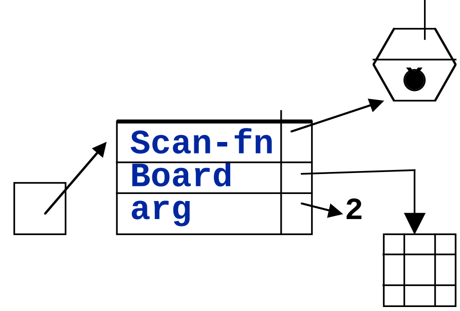
Prof. Louis Steinberg

Lecture 9:

First-order Logic

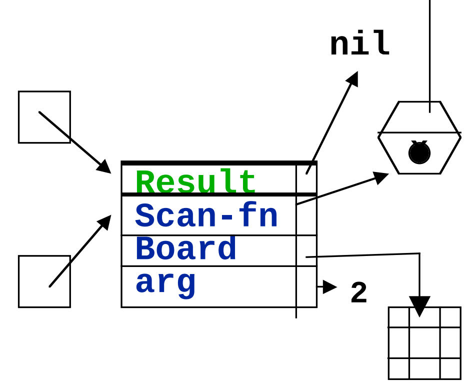
List-scan

```
(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))
```



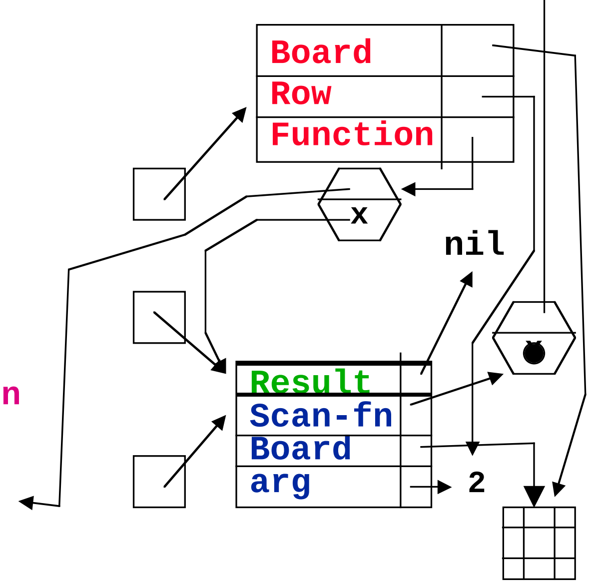
List-scan

```
(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))
```



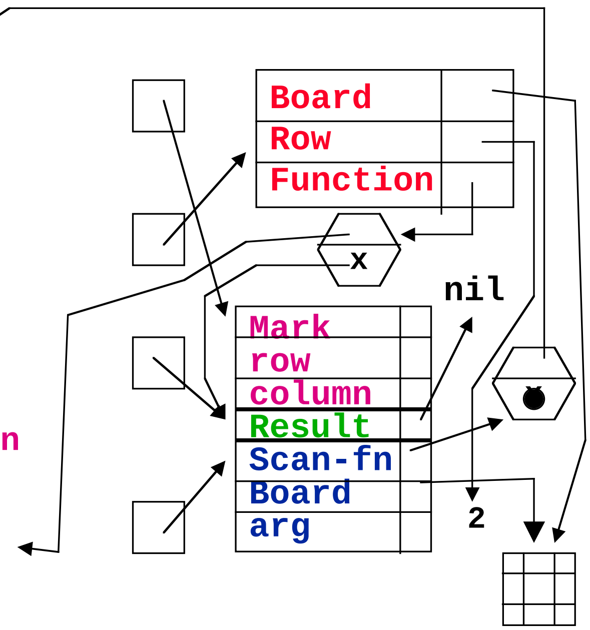
List-scan

```
(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))
```



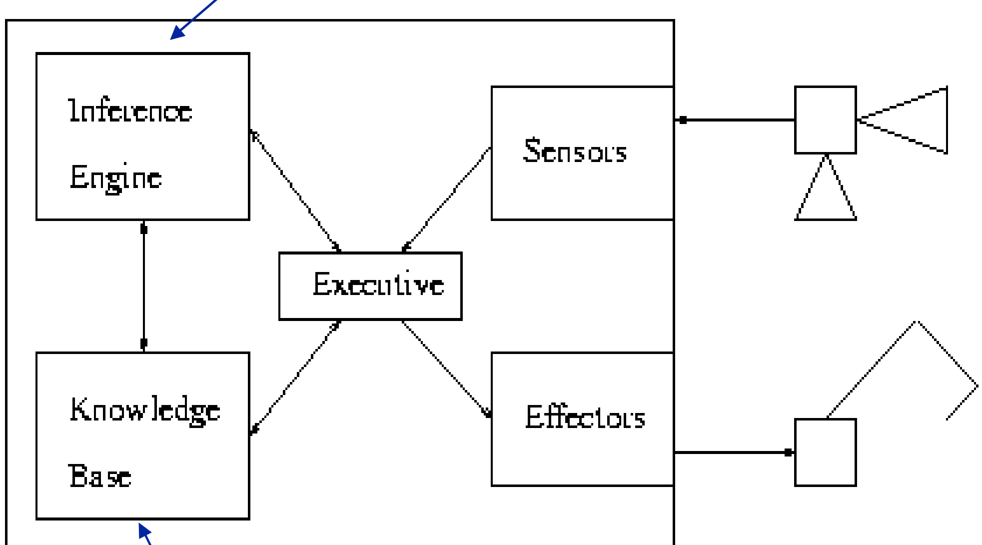
List-scan

```
(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))
```



Knowledge-based Agent Architecture

Derives new facts from old



Stores true facts

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

$$\frac{\alpha \vee \beta \quad \neg\alpha \vee \gamma}{\beta \vee \gamma}$$

There is a pit in (1, 2) or there is a pit in (2, 1).

$\alpha \vee \beta$

There is not a pit in (1, 2) or there is a breeze in (1, 3)

$\neg\alpha \vee \gamma$

---

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

$\beta \vee \gamma$

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:

¬B11 ∨ P12 ∨ P21	**breeze in (1, 1) => pit in (1,2) or (2,1)**
¬P12 ∨ B13	**pit in (1, 2) => breeze in (1, 3)**
B11	**breeze in (1, 1)**
¬B13	**no breeze in (1, 3)**

Proof

$\neg B11 \lor P12 \lor P21$ $\quad$ $B11$

$P12 \lor P21$ $\quad$ $\neg P12 \lor B13$

$P21 \lor B13$ $\quad$ $\neg B13$

$P21$

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 committments 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) ^ Brother(John,Richard)

Older(John,30) => ¬ 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:

- **∀x Cat(x=> Mammal(x))**
- **means ``for any object x, if x is a cat, then x is a mammal.''**

Existential Quantification:

- **∃x Sister(x,Morris) ^ Cat(x)**
- **means ``Morris has a sister who is a cat''**

Nested Quantifiers

- **What do the following mean?**
  - **∀x,y Parent(x,y) => Child(y,x)**
  - **∀x,y Brother(x,y) => Sibling(y,x)**
  - **∀x ∃y Loves(x,y)**
  - **∃y ∀ x Loves(x,y)**

Connections between Quantifiers

$\forall x \neg Likes(x,Morris) \equiv \neg \exists x\ Likes(x,Morris)$

$\exists x \neg Likes(x,Morris) \equiv \neg \forall x\ Likes(x,Morris)$

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**

$(\forall t)\ (\exists p)\ Time(t) \wedge Person(p) \Rightarrow Foolable(p,t)$

$(\exists p)\ (\forall t)\ Time(t) \wedge Person(p) \Rightarrow Foolable(p,t)$

$(\exists t)\ (\forall p)\ Time(t) \wedge Person(p) \Rightarrow Foolable(p,t)$

$(\forall p)(\exists t)\ Time(t) \wedge Person(p) \Rightarrow Foolable(p,t)$

$(\forall t)\ \neg Foolable(Mother(you),t)$

$(\forall p)\ (\forall t)\ \neg Foolable(Mother(you),t)$

First-order logic into English

$(\forall x)$ Hesitates(x) => Lost(x)

$\neg(\exists x)$ Business(x) ^ Like(x,Showbusiness)

$(\forall x)$ Glitters(x) => ¬Gold(x)

$(\exists x)$ Glitters(x) ^ ¬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''.**