CSC 550: Introduction to Artificial Intelligence

Fall 2004

Knowledge representation

- associationist knowledge
  semantic nets, conceptual dependencies
- structured knowledge
  frames, scripts
- alternative approaches

Knowledge representation

underlying thesis of GOFAI: Intelligence requires

- the ability to represent information about the world, and
- the ability to reason with the information

knowledge representation schemes

- logical: use formal logic to represent knowledge
  e.g., state spaces, Prolog databases
- procedural: knowledge as a set of instructions for solving a problem
  e.g., production systems, expert systems (next week)
- associationist: knowledge as objects/concepts and their associations
  e.g., semantic nets, conceptual dependencies
- structured: extend networks to complex data structures with slots/fillers
  e.g., scripts, frames

Semantic nets (Quillian, 1967)

main idea: the meaning of a concept comes from the way it is connected to other concepts

SNOW

in understanding language and/or reasoning in complex environments, we make use of the rich associativity of knowledge

When Timmy woke up and saw snow on the ground, he immediately turned on the radio.

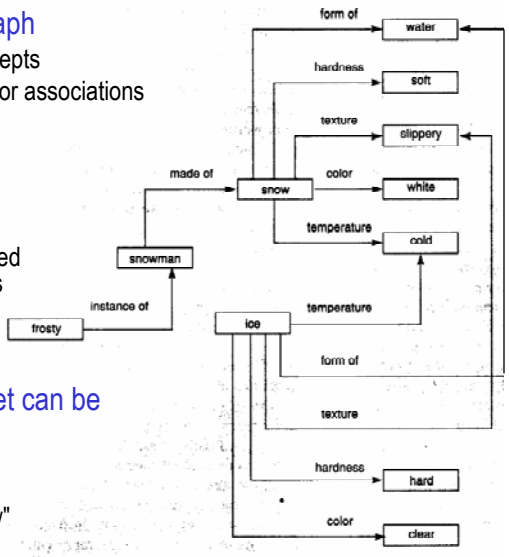
graphs of concepts

can represent knowledge as a graph

- nodes represent objects or concepts
- labeled arcs represent relations or associations

such graphs are known as semantic networks (nets)

- the meaning of a concept is embodied by its associations to other concepts



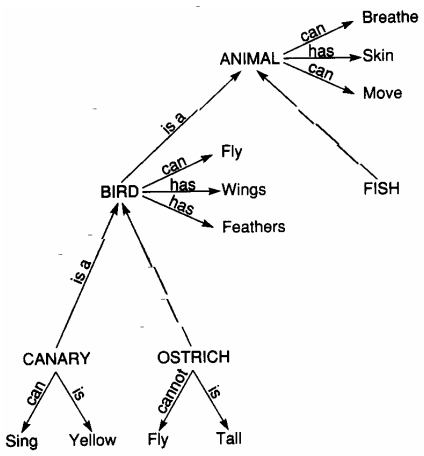
retrieving info from a semantic net can be seen as a graph search problem

to find the texture of snow

1. find the node corresponding to "snow"
2. find the arc labeled "texture"
3. follow the arc to the concept "slippery"

semantic nets & inheritance

in addition to data retrieval, semantic nets can provide for *deduction* using inheritance



since a canary is a bird, it *inherits* the properties of birds (likewise, animals)

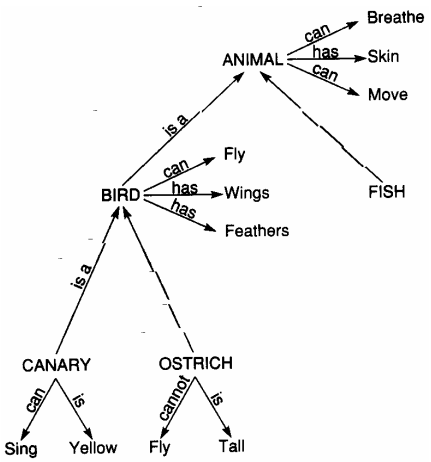
e.g., canary can fly, has skin, …

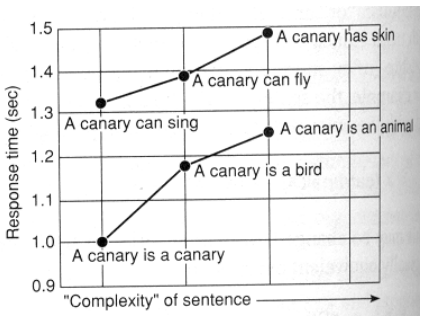
to determine if an object has a property,

- look for the labeled association,
- if no association for that property, follow *is_a* link to parent class and (recursively) look there

Inheritance & cognition

Quillian and Collins (1969) showed that semantic nets with inheritance modeled human information storage and retrieval

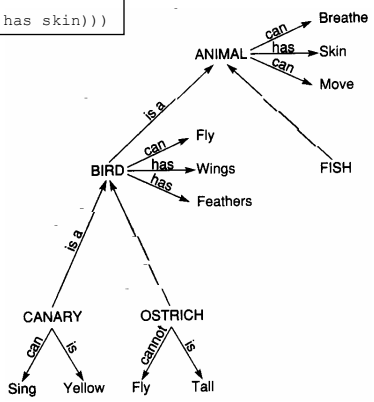




Semantic nets in Scheme

can define a semantic net in Scheme as an association list

```
(define ANIMAL-NET
  '((canary can sing) (canary is yellow) (canary is-a bird)
    (ostrich is tall) (ostrich cannot fly) (ostrich is-a bird)
    (bird can fly) (bird has wings) (bird has feathers)
    (bird is-a animal) (fish is-a animal)
    (animal can breathe) (animal can move) (animal has skin)))
```



Semantic net search

```
;;; net.scm

(define (lookup object property value NETWORK)

  (define (get-parents object NET)
    (cond ((null? NET) '())
          ((and (equal? object (caar NET)) (equal? 'is-a (cadar NET)))
           (cons (caddar NET) (get-parents object (cdr NET))))
          (else (get-parents object (cdr NET)))))

  (define (inherit parents)
    (if (null? parents)
        #f
        (or (lookup (car parents) property value NETWORK)
            (inherit (cdr parents)))))

  (if (member (list object property value) NETWORK)
      #t
     (inherit (get-parents object NETWORK))))
```

to lookup a relation

- if arc with desired label exists, done (SUCCEED)
- otherwise, if is_a relation holds, follow the link and recurse on that object/concept

```
> (lookup 'canary 'is 'yellow ANIMAL-NET)
#t

> (lookup 'canary 'can 'fly ANIMAL-NET)
#t

> (lookup 'canary 'can 'breathe ANIMAL-NET)
#t

> (lookup 'canary 'is 'green ANIMAL-NET)
#f
```

```
> (lookup 'ostrich 'cannot 'fly ANIMAL-NET)
#t

> (lookup 'ostrich 'can 'fly ANIMAL-NET)
#t
```

WHY?

Semantic net search, with negative relations

```
;;; net.scm

(define ANIMAL-NET
  '((canary can sing) (canary is yellow) (canary is-a bird)
    (ostrich is tall) (ostrich (not can) fly) (ostrich is-a bird)
    (bird can fly) (bird has wings) (bird has feathers)
    (bird is-a animal) (fish is-a animal)
    (animal can breathe) (animal can move) (animal has skin)))


(define (lookup object property value NETWORK)

  (define (opposite property)
    (if (symbol? property)
        (list 'not property)
        (cadr property)))

  (define (get-parents object NET)
    (cond ((null? NET) '())
          ((and (equal? object (caar NET)) (equal? 'is-a (cadar NET)))
           (cons (caddar NET) (get-parents object (cdr NET))))
          (else (get-parents object (cdr NET)))))

  (define (inherit parents)
    (if (null? parents)
        #f
        (or (lookup (car parents) property value NETWORK)
            (inherit (cdr parents)))))

  (cond ((member (list object property value) NETWORK) #t)
        ((member (list object (opposite property) value) NETWORK) #f)
        (else (inherit (get-parents object NETWORK)))))
```

to lookup a relation

- if arc with desired label exists, done (SUCCEED)
- if arc with opposite label exists, done (FAIL)
- otherwise, if is_a relation holds, follow the link and recurse on that object/concept

```
> (lookup 'ostrich
          '(not can)
          'fly
          ANIMAL-NET)
#t

> (lookup 'ostrich
          'can
          'fly
          ANIMAL-NET)
#f
```

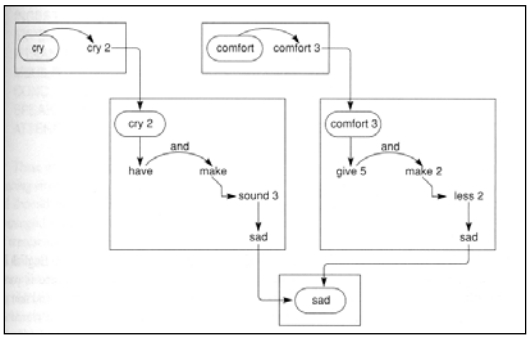

Implementation comments

DISCLAIMER: this semantic net implementation is simplistic

- need to be able to differentiate between instances and classes
- need to differentiate between properties of a class and properties of instances of that class
- need to handle multiple inheritance paths

Quillian used an intersection algorithm to find word relationships

- given two words, conduct breadth first search from each node
- look for common concepts (intersection nodes from the searches)



Conceptual dependency theory

not surprisingly, early semantic nets did not scale well

- most links were general associations
- no real basis for structuring semantic relations

much research has been done in defining richer sets of links

- rely on richer formalism, not richer domain knowledge

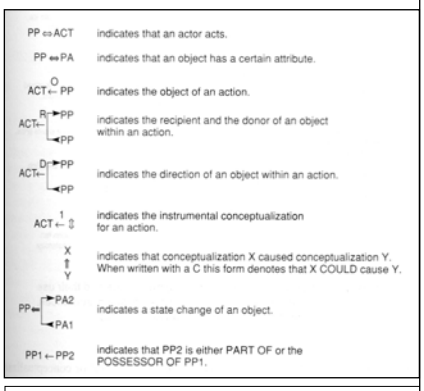
Conceptual Dependency Theory (Schank, 1973)

- attempts to model the semantic structure of natural language
- 4 primitive conceptualizations, from which meaning is built

ACT action
PP objects (picture producers)
AA modifiers of actions (action aiders)
PA modifiers of objects (picture aiders)

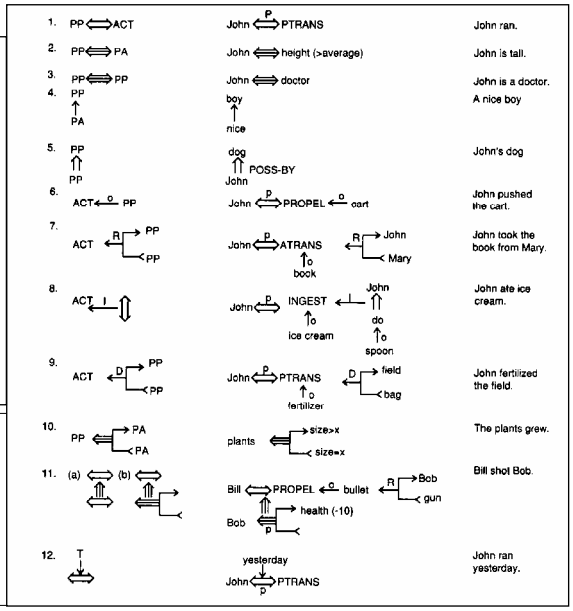
primitive actions include: ATRANS (transfer a relationship, e.g., give)
PTRANS (transfer physical location, e.g., move)
MTRANS (transfer mental information, e.g., tell)
. . .

conceptual dependency relationships

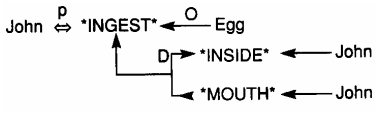


tense/mode modifiers

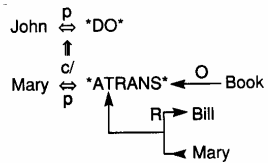
p past
f future
t transition
? interrogative
/ negative
. . .



CD examples



John ate an egg.



John prevented Mary from giving a book to Bill.

CD for natural language understanding

in the context of natural language understanding, the Conceptual Dependency representation has interesting properties:

- knowledge is represented using conceptual primitives
- actual words/phrases are not stored directly
- ideally, representation is independent of the original language (could be English, French, Russian, …)

John sold Mary a book.
Mary bought a book from John.
Mary gave John a check for the book.

these sentences describe the same event – a CD representation would reduce these to the same conceptual symbols

ADVANTAGE: syntax is minimized, semantics matters

RESULT: CD representation is good for understanding or paraphrasing sentences

MARGIE (Schank, 1973)

MARGIE: Memory, Analysis, Response Generation in English

the system combined a

- parser (English → CD)
- generator (CD → English)
- inference engine (inferred info from CD)

MARGIE in inference mode

INPUT: John gave Mary an aspirin.
OUTPUT1: John believes that Mary wants an aspirin.
OUTPUT2: Mary is sick.
OUTPUT3: Mary wants to feel better.
OUTPUT4: Mary will ingest the aspirin.

MARGIE in paraphrase mode

INPUT: John killed Mary by choking her.
OUTPUT1: John strangled Mary.
OUTPUT2: John choked Mary and she died because she could not breathe.

Frames (Minsky, 1975)

in contrast to distributed knowledge networks, can instead organize knowledge into units representing situations or objects

When one encounters a new situation (or makes a substantial change in one's view of a problem) one selects from a memory structure called a "frame." This is a remembered framework to be adapted to fit reality by changing details as necessary.

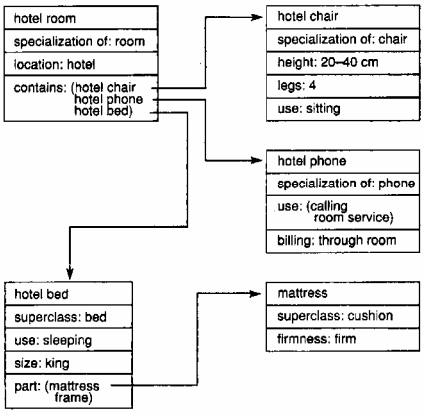
-- Marvin Minsky

HOTEL ROOM

Frame example

a frame is a structured collection of data

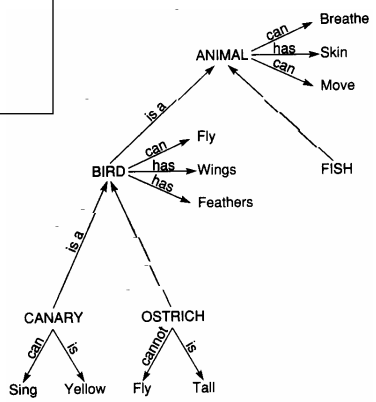
- has *slots* (properties) and *fillers* (values)
- fillers can be links to other frames



Frame set in Scheme

```
(define ANIMAL-FRAME
  '((canary  (can sing)
             (is yellow)
             (is-a bird))
    (ostrich ((not can) fly)
             (is tall)
             (is-a bird))
    (bird    (can fly)
             (has wings feathers)
             (is-a animal))
    (fish    (is-a animal))
    (animal  (can breathe move)
             (has skin))))
```

represent a frame as a nested structure



Frame search

```
;;; frame.scm

(define (lookup object property value FRAME)

  (define (opposite property)
    (if (symbol? property)
        (list 'not property)
        (cadr property)))

  (define (get-parents object)
    (let ((parents (assoc 'is-a (cdr (assoc object FRAME)))))
      (if (not parents)
          '()
          (cdr parents))))

  (define (inherit parents)
    (if (null? parents)
        #f
        (or (lookup (car parents) property value FRAME)
            (inherit (cdr parents)))))

  (let ((entry (assoc object FRAME)))
    (if (not entry)
        #f
        (let ((vals (assoc property (cdr entry)))
              (negvals (assoc (opposite property) (cdr entry))))
          (cond ((and vals (member value (cdr vals))) #t)
                ((and negvals (member value (cdr negvals))) #f)
                (else (inherit (get-parents object))))))))
```

to perform a deduction

get frame information,

- if desired slot exists, get filler
- if opposite of slot exists, fail
- otherwise, if there is an is-a slot, get the parent frame and recurse on that object/concept

Implementation comments

DISCLAIMER: again, this implementation is simplistic

- need to be able to differentiate between instances and classes
- need to differentiate between properties of a class and properties of instances of that class
- need to handle multiple inheritance paths

The structured nature of frames makes them easier to extend

- can include default values for slots
- can specify constraints on slots
- can attach procedures to slots

BASEBALL PLAYER
is_a : athlete
height: 6 ft bats: {left, right, switch} hits : 0 atBats : 0 batting avg: hits/atBats . . .

Representation applications

semantic nets, frames, and scripts were foundational structures

- more recently, they have been adapted and incorporated into hybrid structures

vision

- Minksy saw frames as representing different perspective of an object

understanding

- use frames with defaults to "fill in the blanks" in understanding
  EXAMPLE: "I looked in the janitor's closet …"
- Lenat's AM represented concepts as frames (newly concepts spawned new frames)

smart databases

- Lenat's CYC project used extension of frames, with conventions for slots & fillers
- PARKA project at Maryland uses frame-based language for efficiently accessing large knowledge bases
- Hyundai Engineering uses frame-based system for planning construction projects

interesting note:

- MIT research on frames (and similar research at XEROX PARC) led to object-oriented programming and the OOP approach to software engineering

Scripts (Schank & Abelson, 1975)

a script is a structure that describes a stereotyped sequence of events in a particular context

- closely resembles a frame, but with additional information about the expected sequence of events and the goals/motivations of the actors involved
- the elements of the script are represented using Conceptual Dependency relationships (as such, actions are reduced to conceptual primitives)

EXAMPLE: restaurant script

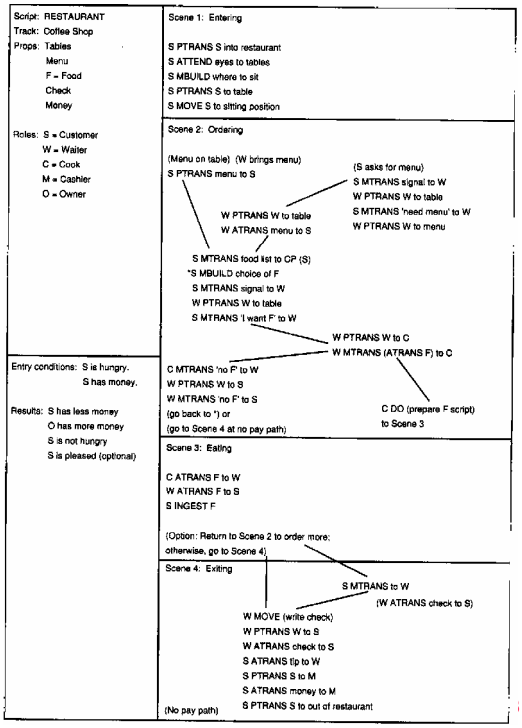
describes: items usually found in a restaurant
people and their roles (e.g., chef, waiter, …)
preconditions and postconditions
common scenes in a restaurant: entering, ordering, eating, leaving

Hotel script

props and roles are identified

pre- and post-conditions

CDs describe actions that occur in each of the individual scenes



Script application

SAM: Script Applier Mechanism

- Cullingford & Schank, 1975

system consisted of:

- parser (extension of MARGIE)
- generator (extension of MARGIE)
- script applier (to check the consistency of the CD repr. with that specified in the script)
- question answerer

INPUT

Friday evening a car swerved off Route 69. The vehicle struck a tree. The passenger, a New Jersey man, was killed. David Hall, 27, was pronounced dead at the scene by Dr. Dana Blanchard, medical examiner. Frank Miller, 32, of 593 Foxon Rd., the driver, was taken to Milford Hospital by Flanagan Ambulance. He was treated and released. No charges were made. Patrolman Robert Onofrio investigated the accident.

ENGLISH SUMMARY

AN AUTOMOBILE HIT A TREE NEAR HIGHWAY 69 FOUR DAYS AGO. DAVID HALL, AGE 27, RESIDENCE IN NEW JERSEY, THE PASSENGER, DIED. FRANK MILLER, AGE 32, RESIDENCE AT 593 FOXON ROAD IN NEW HAVEN, CONNECTICUT, THE DRIVER, WAS SLIGHTLY INJURED. THE POLICE DEPARTMENT DID NOT FILE CHARGES.

SPANISH SUMMARY

UN AUTO CHOCO CONTRA UN ARBOL CERCA DE LA CARRETERA 69 EL VIERNES AL ANOCHECER. DAVID HALL, DE 27 ANOS, QUE VIVIA EN NEW JERSEY, EL PASAGERO, SE MURIO. FRANK MILLER, DE 32 ANOS, QUE VIVIA EN 593 CALLE FOXON EN NEW HAVEN, CONNECTICUT, EL CHOFER, QUEDO UN POCO HERIDO. EL DEPARTAMENTO DE POLICIA NO ACUSO A NADIE.

QUESTION-ANSWERING OUTPUT

Q1: Was anyone killed?
A1: YES, DAVID HALL DIED.

Q2: Was anyone hurt?
A2: YES, FRANK MILLER WAS SLIGHTLY INJURED.

Q3: Why was Miller hurt?
A3: BECAUSE THE AUTOMOBILE HIT THE TREE.

Q4: Did Miller go to the hospital?
A4: YES, HE WAS IN THE MILFORD HOSPITAL.

Q5: How did Miller get to the hospital?
A5: THE FLANAGAN AMBULANCE COMPANY TOOK HIM TO THE MILFORD HOSPITAL.

Alternatives to explicit representation

connectionist & emergent approaches (later)

Subsumption architecture (Brooks, MIT)

- claim: intelligence is the product of the interaction between an appropriately layered system and its environment
- architecture is a collection of task-handling behaviors, with each behavior accomplished via a finite state machine
- limited feedback between layers of behavior
- "… in simple levels of intelligence, explicit representations and models of the world simply get in the way. It turns out to be better to use the world as its own model." (Brooks)

Copycat architecture (Mitchell & Hofstadter, Indiana)

- builds on representation techniques from semantic nets, blackboards, connectionist networks, and classifier systems
- supports semantic net-like representation that can evolve
- emphasizes analogical reasoning

Next week…

Expert systems

- rule-based vs. model-based vs. case-based
- probabilistic vs. fuzzy reasoning

Read Chapters 7, 8

Be prepared for a quiz on

- this week's lecture (moderately thorough)
- the reading (superficial)