Structured Knowledge Representations: Frames, Scripts and Situation Models

Intelligence: Knowledge and Understanding

Structured Knowledge Representation

Relations
Relations as N-Ary Predicates
Implicit vs Explicit representations for Relations

Schema Systems

Frames
Situation Models
Scripts
Fundamental problems with Schema Systems
Frames for Describing Blocks World

Bibliography:


**Intelligence: Knowledge and Understanding**

Question from séance 1: What do we mean by Intelligence?

Intelligence is the ability to Know and to Understanding.

*The Turing Test*

![Turing Test Diagram]

From Turing - Intelligence is a property of an agent.

To be considered as intelligent an agent must be

1) Embodied - Able to act  
2) Autonomous - Able to act to maintain integrity  
3) Situated - The ability to act appropriate to the domain and task; Rational

We are interested in technologies that allow a program to behave with intelligence.

**Knowledge**

Knowledge is ability; The ability to do things; The ability to act to accomplish goals

**Meta-Knowledge: Learning and Reasoning:**

Learning and reasoning are forms of Meta-Knowledge.  
Meta-knowledge: The ability to generate new knowledge (new abilities)  
Learning: Acquisition of knowledge from training and experience  
Reasoning: The acquisition of knowledge from understanding.

What does it mean to understand? The ability to predict and explain.

Reasoning generates new from understanding (The ability to predict and explain):

Heuristic for understanding (reasoning); (tools to predict and explain)  
The 6 Ws - What, Where, Why, When, Who, How  
In English this is really 5W1H
Structured Knowledge Representation

Structured Knowledge Representations are declarative structures for reasoning and communication.

Structured knowledge representations (Schema) are tools for answering the 5W1H and variations.

A key property of Structured Representations (schema) is the association of facts with procedures for reasoning. Entities are represented by objects, with properties represented in “slots”. Procedures for reasoning about entities are represented by methods (procedures), associated directly with objects. Objects are organized in a class hierarchy with both slots and procedures inherited within the hierarchy.

Today we will review 3 forms of schema: Frames, Situation Models, Scripts.

Situations Models: Where, Why
Scripts: When, How

We will use a simplified “Blocks World” to provide examples.

Relations
As we have seen, a fundamental concept for organizing such structures is the concept of "relation". Examples include temporal relations, spatial relations, family relations, social relations, administrative organizations, military hierarchies, etc.

Relations are formalized as Predicates (Truth functions). A predicates is function that assigns a property to an association of arguments. Normally, predicates are assumed to be Boolean functions, but an interesting research problem is how to use probabilistic predicates to represent relations.

A predicates is a function that tells whether or not a relation is valid for a set of entities. Classically, predicates are treated as Boolean functions that can only return a value of TRUE or FALSE. As we have seen, in probabilistic reasoning, predicates represent the likelihood that the relation holds, with a value between 0 and 1.
Structured Knowledge Representation

Lesson 18

Relations as N-Ary Predicates

Relations have Arity

The "Arity" of a relation is the number of arguments. Arity represents the number of entities associated by the relation. Relations may have an arity of 0 or more arguments.

The valence or Arity of a relation is the number of entities that it associates.

Unary: Man(Bob) ;; Bob is a human of gender male
Binary Brother(Bob, Chris) ;; Bob and Chris are brothers
Ternary Triangle(A, B, C) ;; a geometric relation associating points or lines
Variadic: Set(A, B, C, D) ;; Collections, assemblies, etc

In some systems it is possible to have functions with variable arity. These are called polyadic functions or variadic functions.

Implicit vs Explicit representations for Relations

Relations can be represented "implicitly" or "explicitly".

Implicit Representations
With an implicit representation, the relation is represented as a slot. We saw this with the family relations:

(defclass PERSON (slot NAME) (slot FATHER) (slot MOTHER) ).

The slot FATHER contains a pointer to an object of the class PERSON that represents the father of the person. The pointer is the object address.

Implicit representations are simple and more efficient in computing and memory.
However, an implicit (slot based) representation for relations is not easily completed with meta information, and does not easily support generalization for reasoning and for natural language interaction.

When convenient, we will use implicit relations for simplicity. However, some forms of reasoning are much easier with an explicit representation.

**Explicit representation**

With an explicit representation, a relation is represented by a separate object.

A slot holds a pointer to the object that represents the relation. This object can then provide additional information about the object, such as what, where, why, when, who and how.

With an explicit representation for relations, it is possible to write a set of general procedures for acquiring (learning), reasoning, and explaining that apply to all relations.

With an implicit representation, such procedures would be specific to each class.

Thus explicit relations support generalized methods acquiring (learning), reasoning, and explaining about relations.

Examples:

Temporal Reasoning. Allen’s temporal reasoning is much easier to program using explicit models of relations.

Is-a Relations for reasoning about classes and categories.
Structured knowledge representations, or Schema, are used to represent problem spaces for reasoning, perception, problem solving and natural language interaction. In the following we will examine 3 classic structured knowledge representations: Frames, Scripts, and Situation Models. All of these are examples of Schema systems.

Frames
M. Minsky proposed Frames as a structure to guide scene interpretation in computer vision.
In computer vision, the problem can be stated as transforming an array of numbers into a symbolic description of a scene. This problem can be made easy when you know what to look for and how to look for it.

Minsky intended Frames to guide interpretation in a top down manner, telling a vision system where to look and what to look for. Minsky's insight was that it is much easier to see if you know what to look for.

A frame is composed of a collection of entities (objects) joined by relations (predicates), and a collection of procedures for perceiving and reasoning about entities and their relations.

Common relations include class hierarchies (ISA, AKO) and part hierarchies (PART-OF and COMPOSED-OF), Spatial relations (left-of, right-of, above, below).

ISA represents class hierarchy for entities.
Note that is-a relations are now commonly built in to object oriented programming tools. However, when interpreted, is-a can also be used for reasoning.

For example, we can write rules that recover the class hierarchy.

(defrule find-blocks
   (object (is-a ?c) (NAME ?n))
=>
   (printout t "Object " ?n " is a " ?c crlf)

AKO stands for A Kind Of. AKO is used to designate an entity that serves as an example for default reasoning.)
Frames provide visual context to guide scene interpretation. Typically, a Frame describes a class of entity such as a CHAIR or ROOM. A Frame tells the program what to look for and where to look for it.

They ask allow a system to ask questions about objects, such as what and where

Frames are formalized as a set of relations between objects. Relations can be implicit or explicit. Slots can hold pointers for other frames (implicit representation) or hold pointers to relations that refer to other entities (explicit representation).

Relations represents information about the object, such as part relations (composed of, part-of), Position relations (above, below, beside, inside, contains), Time relations (before, after, during), as well as specific properties of the entity (size, position, color, orientation). Ultimately, some slots point to raw perceptions.

When a slot points to an entity, the entity is said to play a "role" in the frame. Frames typically come with methods (procedures) for searching for the entities that can plays roles in the frame. Typically a slot-filling procedure will apply a set of acceptance tests to an entity to see if it satisfies the requirements for the role.
Frames generally include typical examples (prototypes) that can serve as examples in reasoning, and default values that are used if no entity has been found to fill the slot. Thus frames can be used for abstract reasoning or for reasoning when perception is not possible.

The term Frames has come to represent any general representation of common sense knowledge using a slot and filler structure. Slots tell what entities to find and fillers are procedures to find entities that can fill slots.

A system of frames for visual observation of a cube.
(from E. Rich “Artificial Intelligence”, Fig 7-13, p231)

Discovering the appropriate frame for reasoning is called “The Frame Problem”.

18-9
Situation Models


Situations models are used in cognitive psychology to express the mental models that people use to understand and reason.

Properties: Sensed values
Entities: Anything that can be named or designated; People, things, etc.
(Entities can be defined as a correlation of properties)
Relations: An N-ary predicate (N=1,2,3 …) over entities.
(Relations are defined as tests on the properties of entities).
Situation: A set of relations between entities

Example: John is facing Mary. John is talking to Mary. Mary is not listening.

facingP(John, Mary) ^ talkingP(John, Mary) ^ not listeningP(Mary, John)

Situations can be organized into a state space referred to as a situation network. Each situation (or state) corresponds to a specific configuration of relations between entities. A change in relation results in a change in situation (or state).

The situation graph, along with the set of entities and relations is called a Context.

Each situation can prescribe and proscribe behaviours.
1) Behaviors: List of actions and reactions that are allowed or forbidden for each situation. Behaviors are commonly encoded as Condition-Action rules.
2) Attention: entities and relations for the system to observe, with methods to observe the entities
3) Default values: Expectations for entities, relations, and properties
4) Possible situations: Adjacent neighbors in the situation graph.
Each situation indicates:

 Transition probabilities for next situations
 Transition probabilities for next situations
 The appropriateness or inappropriateness of behaviors

Behaviors include

  1) methods for sensing and perception, and
  2) appropriateness of actions
  3) changes in state in reaction to events.

The sets of entities, relations, behaviors, and situations define a "Context".

Situation models are used to construct context aware systems.

A "Context" is defined as

  1) A set of entities, with their properties.
  2) A set of relations between entities
  3) A network of situations, such that each situation specifies
     - A list of adjacent situations, possibly with transition probabilities.
     - A list of system behaviors that are allowed or forbidden, possibly with preferences (appropriateness) for the situation.
Scripts

A script is a structure used to represent a stereotypical sequence of events. Scripts are used for interpreting stories. For examples, scripts used to construct systems that interpret and extract information from Newspaper Stories.

Scripts are also used to observe an actor and to describe (or recognize) what the actor is doing. This includes plan-recognition as well as activity description.

Finally, scripts can be used to represent procedural knowledge for plans.

A script is composed of
1) Props: Entities (objects) involved in the script.
2) Scenes: A set of situations (states) of the entities.
3) Roles: Actors (agents) that can provoke changes in the scenes. (Changes in situation). Actors are typically people, but may be artificial.
4) Tracks: Specific sequences of scenes that may occur depending on the actions of the actors.
5) Entry: The initial scene
6) Result: The final scene.

The script can be represented as a tree or network of scenes, driven by actions of the actors (rolls).

Example of a script
The classic example is the restaurant script:
Props: A restaurant with an entrance, tables, chairs, plates, eating utensils, glasses, menu, etc
Actors: The host (Maitre d'Hotel), clients, servers, chef, bus-boy, etc.
Scenes: Entry, seating, reading the menu, ordering, serving, requesting the check, paying, leaving, etc.

Scripts provide context for default reasoning.
As with Frames, scripts drive interpretation by providing procedures that tell a system what to look for and where to look for it.

Scripts also provide default knowledge for reasoning about stories or actions. For example, For story understanding, the story will typically only provide sparse detail of what happened. The reader is expected to fill in the missing knowledge with default knowledge.
Structured Knowledge Representation

Lesson 18

**Fundamental problems with Schema Systems.**

Frames, Scripts and Situation Models are examples of Schema systems.

Some Fundamental Problem with all schema systems.

1) Knowledge Acquisition: Learning a schema system is long, tedious, and ad hoc process. There is a temptation to overload with useless information.

Research Challenge: Learning, Development and Adaptation.

2) Context Recognition (The Frame problem): Many problems are easily solved when the context and situation are known. Recognizing the correct context can be very difficult.

Research Challenge: Context recognition.

3) Ontology: Two schema systems describing the same context, may not use the same symbols for entities and relations. Communication requires a shared ontology.

Research Challenge: Semantic Alignment.

Alignment is the problem of determining equivalences in symbol expressions.

Solution these can be obtained from probabilistic techniques for learning and recognition.
Frames for Describing Blocks World.

We can use blocks world as a source of examples for Schema.

Blocks world is an abstract, toy world for exploring problems.
Blocks world is a "Closed" world. It has a finite number of states.
Blocks world is composed of a finite number of blocks in a finite number of states.

Blocks world is composed of:
- A set of cubic blocks
- A table
- A set of spatial relations between blocks and the table
- A robot hand that can move blocks.

Minsky tried used Frames as a model for Control of Perception for Computer vision.

Frames for describing Blocks World

(defclass TABLE (is-a USER)
  (role concrete)
  (slot NAME (create-accessor read-write))
  (slot ISA (default table)(create-accessor read-write))
)

(defmessage-handler TABLE What-is ()
  (printout t ?self:NAME " is a " ?self:ISA crlf)
)

(make-instance [T] of TABLE (NAME T)(ISA Table))
(send [T] get-NAME)
(send [T] What-is)

(defclass BLOCK (is-a USER)
  (role concrete)
  (slot NAME (create-accessor read-write))
  (slot ISA (default table)(create-accessor read-write))
  (slot NFACES (default 3)) ;; number of faces
  (slot FRONT (create-accessor read-write))
  (slot SIDE (create-accessor read-write))
  (slot TOP (create-accessor read-write))
  (slot ISA (default Block))
)

(defmessage-handler ENTITY What-is ()
  (printout t ?self:NAME “ is a “ ?self:ISA CRLF)
)
(make-instance [B] of BLOCK)

(defclass POLYGON (is-a ENTITY)
    (role abstract)
    (slot N-SIDES)
)

(defmessage-handler POLYGON WHAT-IS ()
    (printout t ?self:NAME " is a " polygon " composed of " ?self:N-SIDES " line segments." CRLF)
)

(defclass FACE (is-a POLYGON)
    (slot N-SIDES (default 4)) ;; edges
    (slot X) ; Horizontal Center
    (slot Y) ; Vertical center
    (slot P-LINE-1) ;; parallel relations
    (slot P-LINE-2) ;; parallel relations
)

(defmessage-handler FIND-CUBE ()
    (bind ?SELF:FRONT (make-instance of FACE)
    (bind ?SELF:SIDE (make-instance of FACE)
    (bind ?SELF:TOP (make-instance of FACE)
    (find-p-lines) ;; procedure to find parallel lines.
    )
)

(deffunction find-face
    ?f <- (object (is-a FACE) (x nil) (y nil)
    ?pl1 <- (object (is-a
    =>
    (assert (find-p-lines))
    )
)