KR Introduction

- General problem in Computer Science
- Solutions = Data Structures
  - words
  - arrays
  - records
  - list
- More specific problem in AI
- Solutions = knowledge structures
  - lists
  - trees
  - procedural representations
  - logic and predicate calculus
  - rules
  - semantic nets and frames
  - scripts

Kinds of Knowledge

Things we need to talk about and reason about; what do we know?

- Objects
  - Descriptions
  - Classifications
- Events
  - Time sequence
  - Cause and effect
- Relationships
  - Among objects
  - Between objects and events
- Meta-knowledge

Distinguish between knowledge and its representation

- Mappings are not one-to-one
- Never get it complete or exactly right

Types of Knowledge

- a priori knowledge
  - comes before knowledge perceived through senses
  - considered to be universally true
- a posteriori knowledge
  - knowledge verifiable through the senses
  - may not always be reliable
- procedural knowledge
  - knowing how to do something
- declarative knowledge
  - knowing that something is true or false
- tacit knowledge
  - knowledge not easily expressed by language

Characteristics of a good KR:

- It should
  - Be able to represent the knowledge important to the problem
  - Reflect the structure of knowledge in the domain
    - Otherwise our development is a constant process of distorting things to make them fit.
  - Capture knowledge at the appropriate level of granularity
  - Support incremental, iterative development
- It should *not*
  - Be too difficult to reason about
  - Require that more knowledge be represented than is needed to solve the problem

Structured Knowledge Representations

- Modeling-based representations reflect the structure of the domain, and then reason based on the model.
  - Semantic Nets
  - Frames
  - Scripts
- Sometimes called associative networks

Basics of Associative Networks

- All include
  - Concepts
  - Various kinds of links between concepts
    - “has-part” or aggregation
    - “is-a” or specialization
    - More specialized depending on domain
- Typically also include
  - Inheritance
  - Some kind of procedural attachment

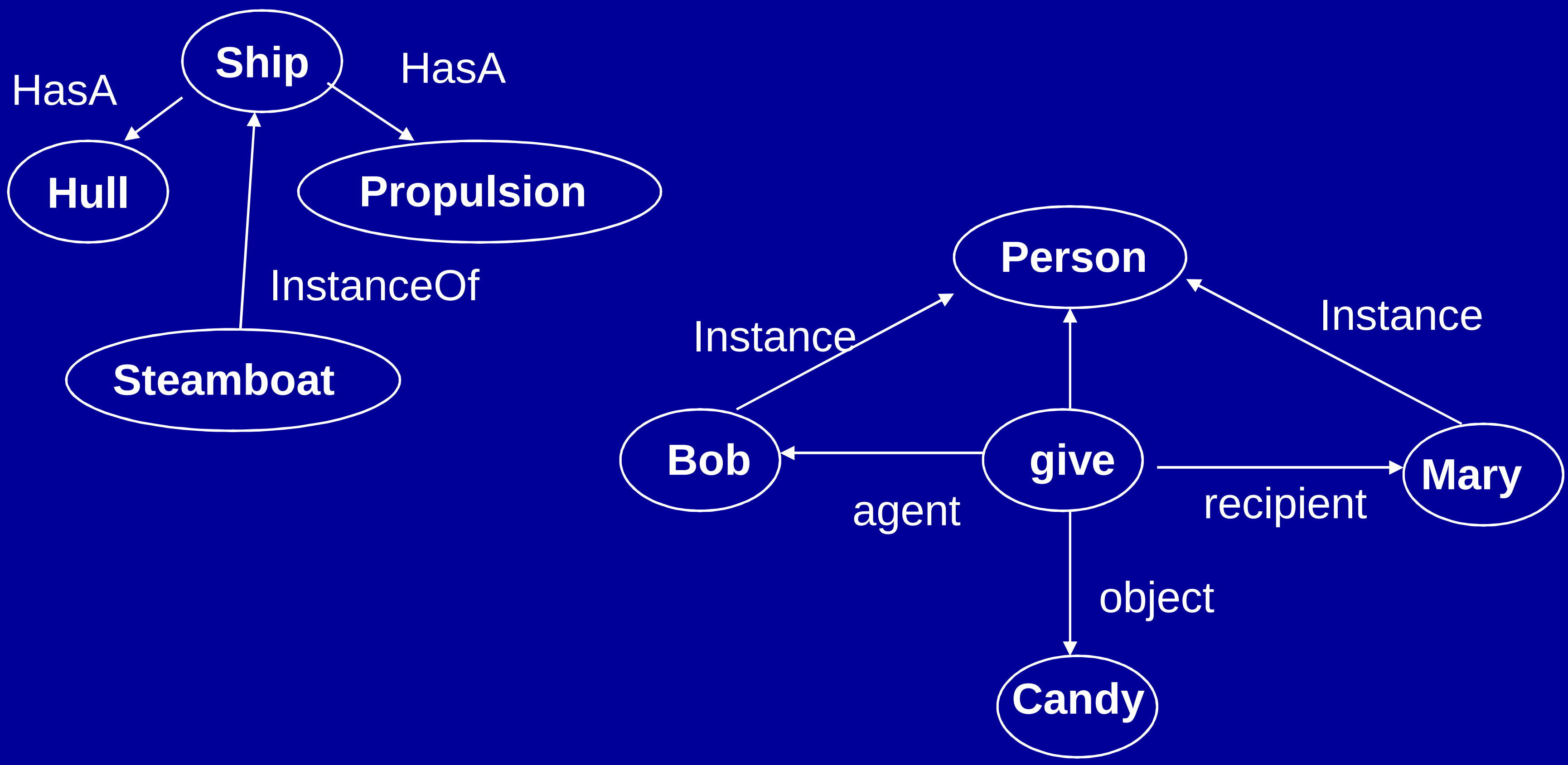
Semantic Nets

- graphical representation for propositional information
- originally developed by M. R. Quillian as a model for human memory
- labeled, directed graph
- nodes represent objects, concepts, or situations
  - labels indicate the name
  - nodes can be instances (individual objects) or classes (generic nodes)
- links represent relationships
  - the relationships contain the structural information of the knowledge to be represented
  - the label indicates the type of the relationship

Semantic Nets Components

- Nodes or Concepts
- Arcs or Links
- Inheritance
- Generic (class) and Individual (object)
- Constraints, procedural attachments
- Nodes
  - "things" or "objects" or "concepts". Typically a demonstrative entity, a noun.
  - E.g., ship, computer, day, dream
- Arcs
  - relationships. Typically a few types expressing important relationships among nodes. The type carries semantic information.
  - E.g., is_a, has_part, provides_power_to, agent

Semantic Net Examples

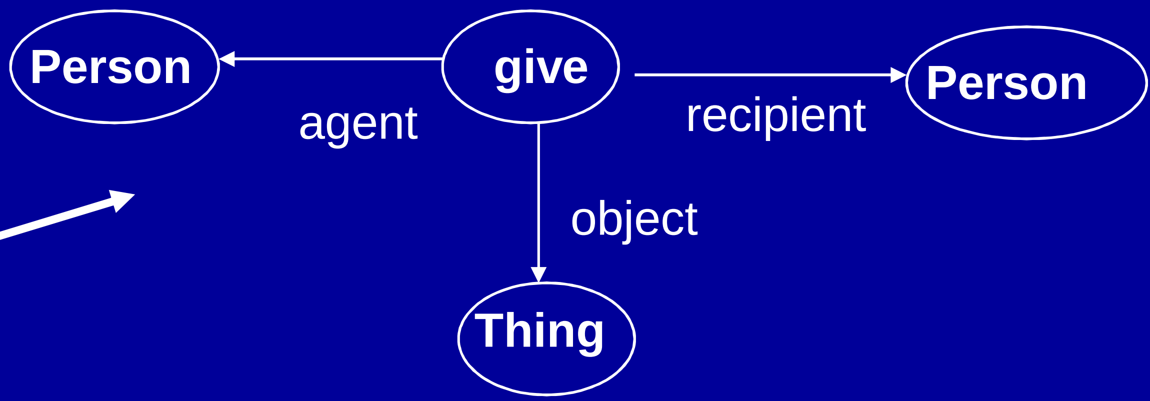


Generic/Individual

- Generic describes the idea--the “notion”
  - static
- Individual or instance describes a real entity
  - must conform to notion of generic
  - dynamic
  - “individuate” or “instantiate”
- Some representations distinguish two kinds of links:
  - Instance Of: instance level relationship
  - A Kind Of: class level relationship
- ISA can mean either, depending on whose terminology you are following ☹.

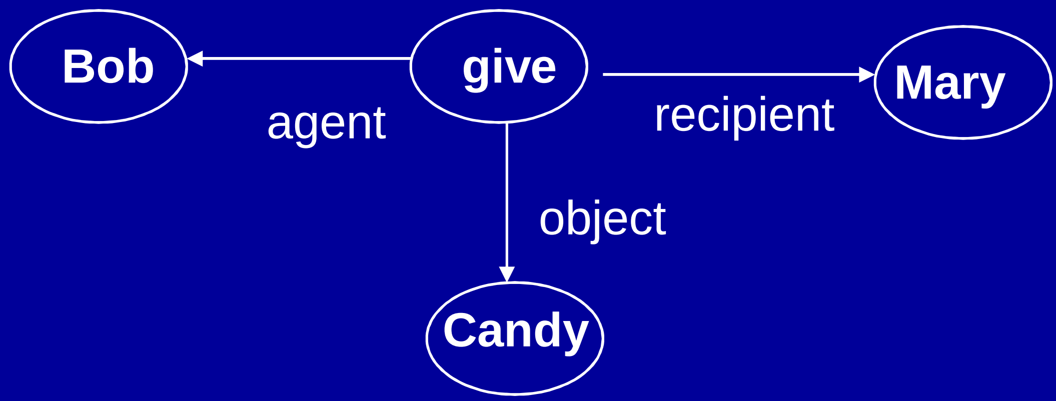
Individuation example

Generic Representation



Process the sentence
“Bob gave Mary some candy”.

Instantiation



Relationships

- without relationships, knowledge is an unrelated collection of facts
  - reasoning about these facts is not very interesting
- relationships express structure in the collection of facts
  - this allows the generation of meaningful new knowledge
    - generation of new facts
    - generation of new relationships

Types of Relationships

- relationships can be arbitrarily defined by the knowledge engineer
  - allows great flexibility
  - for reasoning, the inference mechanism must know how relationships can be used to generate new knowledge
    - inference methods may have to be specified for every relationship
- frequently used relationships
  - IS-A
    - relates an instance (individual node) to a class (generic node)
  - AKO (a-kind-of)
    - relates one class (subclass) to another class (superclass)

Inheritance

- Inheritance is the duplication of a concept's relationships by its descendents.
  - Typically follows specialization hierarchy.
  - Not all relationships are inherited
- In semantic nets, inherited relationships may be further constrained but are not typically overridden.
  - Mammals have hair. A whale is a mammal. We can't say whales don't have hair, but we can further constrain it to say that they have very sparse hair.
- Efficient for creation, maintenance and storage
- Inefficient for reasoning

Schemata

- suitable for the representation of complex knowledge
  - causal relationships between a percept or action and its outcome
  - nodes can have an internal structure
  - for humans often tacit knowledge
- related to the notion of *records* in computer science

Concept Schema

- abstraction that captures general/typical properties of objects
  - has the most important properties that one usually associates with an object of that type
    - may be dependent on task, context, background and capabilities of the user, …
  - similar to stereotypes
- makes reasoning simpler by concentrating on the essential aspects
- may still require relationship-specific inference methods

Schema Examples

- the most frequently used instances of schemata are
  - frames [Minsky 1975]
  - scripts [Schank 1977]
- frames consist of a group of slots and fillers to define a stereotypical objects
- scripts are time-ordered sequences of frames

Frame

- represents related knowledge about a subject
  - provides default values for most slots
- frames are organized hierarchically
  - allows the use of inheritance
- knowledge is usually organized according to cause and effect relationships
  - slots can contain all kinds of items
    - rules, facts, images, video, comments, debugging info, questions, hypotheses, other frames
  - slots can also have *procedural attachments*
    - procedures that are invoked in specific situations involving a particular slot
      - on creation, modification, removal of the slot value

Simple Frame Example

Slot Name	***Filler***
name	Astérix
height	small
weight	low
profession	warrior
armor	helmet
intelligence	very high
marital status	presumed single

Overview of Frame Structure

- two basic elements: *slots* and *facets* (fillers, values, etc.);
- typically have parent and offspring slots
  - used to establish a property inheritance hierarchy
    (e.g., specialization-of)
- descriptive slots
  - contain declarative information or data (static knowledge)
- procedural attachments
  - contain functions which can direct the reasoning process (dynamic knowledge)
    (e.g., "activate a certain rule if a value exceeds a given level")
- data-driven, event-driven (bottom-up reasoning)
- expectation-drive or top-down reasoning
- pointers to related frames/scripts - can be used to transfer control to a more appropriate frame

Slots

- each slot contains one or more facets
- facets may take the following forms:
  – Explicit values
  – Default -- used if there is not other value present
  – Range -- what kind of information can appear in the slot
  – If-added -- procedural attachment which specifies an action to be taken when a value in the slot is added or modified (data-driven, event-driven or bottom-up reasoning)
  – If-needed -- procedural attachment which triggers a procedure which goes out to get information which the slot doesn't have (expectation-driven; top-down reasoning)
  – Other -- may contain frames, rules, semantic networks, or other types of knowledge

Usage of Frames

- filling slots in frames
  - can inherit the value directly
  - can get a default value
  - these two are relatively inexpensive
  - can derive information through the attached procedures (or methods) that also take advantage of current context (slot-specific heuristics)
  - filling in slots also confirms that frame or script is appropriate for this particular situation

Example Frame: Low-level frame

Object	This AI class
ISA	AI class
Date	Fall, 2005
Time	6:15-9PM Thursdays
Instructor	Matuszek
Enrollment	12
Has_A	ROSTER

Frame Example: default frame

Object	AI Class
ISA	Class
Date	
Time	*6:15-9AM*
Instructor	IF-NEEDED: Ask department IF-ADDED: Update payroll
Enrollment	Count ROSTER: Student-IDs
Has_A	ROSTER

Another Example: high-level frame

Object	ROSTER
ISA	Class Record
Enrollment Limit	25
Enrollment	
Status	*Open*
Student-IDs	IF-ADDED Increment Filler(Enrollment) If Filler(Enrollment) = Filler(Enrollment Limit) Then Filler(Status) = Closed IF-DROPPED

Frames vs Semantic Nets

- Frames and nets capture comparable knowledge
- You can automatically transform a frame into a net and vice versa
- Differences are more in typical usage:
  – Semantic nets are normally considered as *specifications*, and do not allow exceptions or defaults
  – Frames are normally considered as *typical* descriptions; *defaults and overrides* are expected
- Nets typically distinguish strongly between classes and instances; frames typically do instantiation at the slot level and don’t have a clearcut distinction at the frame level
- Which is preferable depends on your domain!