Overview of Knowledge Representation and Reasoning

Tim Finin
University of Maryland
Baltimore County

January 2002

Some material adapted from Richard Fikes, Stanford.

Questions

- What’s the difference between data, information and knowledge?
  - Intensional vs. extensional information?
  - Particular vs. general information?
- What does it mean to know something?
  - Philosophers often define knowledge as “justified, true belief”
  - Early AI scientists considered appropriate use of knowledge to be a key
- How does our way of conceptualizing the world influence the way we think and act.
Knowledge Representation

“We base ourselves on the idea that in order for a program to be capable of learning something it must first be capable of being told it.

We shall therefore say that a program has common sense if it automatically deduces for itself a sufficiently wide class of immediate consequences of anything it is told and what it already knows.”


http://www-formal.stanford.edu/jmc/mcc59.html

Knowledge Representation

- Representation of knowledge
  - Description of world of interest
  - Usable by machine to make conclusions about that world

- Intelligent System
  - Computational system
  - Uses an explicitly represented store of knowledge
  - To reason about its goals, environment, other agents, itself

- Reasoning based on explicitly represented knowledge

- Working hypothesis
  - Knowledge of the world -
    - Can be articulated
    - Used as needed
Sample Issues in KR

- What form is the knowledge to be expressed?
- How can a reasoning mechanism generate implicit knowledge?
- How can knowledge be used to influence system behavior?
- How is incomplete or noisy information handled?
- How can we represent and reason?
- How can practical results be obtained when reasoning is intractable?

Knowledge Representation and Reasoning

- Focus on designing forms for expressing information
  - Mostly “general-purpose” languages & systems
  - Must consider reasoning to be done with representation
- Expressiveness vs. tractability tradeoff
  - How to express what we know
  - How to reason with what we express
  - “A Fundamental Tradeoff in Knowledge Representation and Reasoning”
    - H. Levesque, R. Brachman; in Readings in Knowledge Representation; R. Brachman and H. Levesque (eds); Morgan Kaufman; 1985.
KR and Data Base Research

- Both "represent" knowledge
- Data bases contain only "ground literals"
  - No disjunctions
  - No quantifiers
- Data base schema provide quantified information
- Deductive data bases include implications
- Data base concerns -
  - Efficient access and management of large data bases
  - Concurrent updating
- KR concerns -
  - Expressivity
  - Effective reasoning

Early History of KR ('60's - '70's)

- Origins
  - Problem solving work at CMU and MIT
  - Natural language understanding
  - Many ad hoc formalisms
  - "Procedural" vs. "declarative" knowledge
- No formal semantics
- Problems:
  - How do we assign "meaning"
  - How can we say that a computer "understands"?
Emerging Paradigms (‘70’s - ‘80’s)

- Semantic nets
  - Unstructured node-link graphs
  - No semantics to support interpretation
  - No axioms to support reasoning
  - “What’s in a Link: Foundations for Semantic Nets”

- Frames
- Production rules
- Predicate logic

Nodes and Arcs

- arcs define binary relationships which hold between objects denoted by the nodes.
Semantic Networks
- The ISA (is a) or AKO (a kind of) relation is often used to link a class and its superclass.
- And sometimes an instance and its class.
- Some links (e.g. haspart) are inherited along ISA paths.
- The semantics of a semantic net can be relatively informal or very formal (often defined at the implementation level).

Reification
- Non-binary relationships can be represented by “turning the relationship into an object”
- This is an example of what logicians call “reification”
  - reify v : consider an abstract concept to be real
- We might want to represent the generic give event as a relation involving three things: a giver, a recipient and an object, give(john, mary, book32)
Individuals and Classes

Many semantic networks distinguish:
- nodes representing individuals and those representing classes
- the “subclass” relation from the “instance-of” relation

Emerging Paradigms (‘70’s - ‘80’s)

- Semantic nets
- Frames
  - Structured semantic nets
  - Object-oriented descriptions
  - Prototypes
  - Classsubclass taxonomies
- “A Framework for Representing Knowledge”
  - M. Minsky; in Mind Design; edited by J. Haugeland; MIT Press; 1981.
- Production rules
- Predicate logic
From Semantic Nets to Frames

- Semantic networks morphed into Frame Representation Languages in the 70’s and 80’s.
- A Frame is a lot like the notion of an object in OOP, but has more meta-data.
- A frame has a set of slots.
- A slot represents a relation to another frame (or value).
- A slot has one or more facets.
- A facet represents some aspect of the relation.

Facets

- A slot in a frame holds more than a value.
- Other facets might include:
  - current fillers (e.g., values)
  - default fillers
  - minimum and maximum number of fillers
  - type restriction on fillers (usually expressed as another frame object)
  - attached procedures (if-needed, if-added, if-removed)
  - salience measure
  - attached constraints or axioms
- In some systems, the slots themselves are instances of frames
Example Class-Subclass Taxonomy

Class Frame

Class Edited-Book
- Defined in Ontology: Documents
- Source code: documents.top

Description: An edited book is a book whose authors are known as editors.

Instances Of:
- Class: Document, Set
- Subclass Of: Book, Document, Individual, Individual Thing, Thing

Slots:
- Has Author:
  - MinCardinality = 1
- Has Editor:
  - MinCardinality = 1
- Publication Date:
  - MinCardinality = 1
Example Instance Frame

<table>
<thead>
<tr>
<th>Instance Solving-Frame-Problem</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Defined in Ontology: My documents</td>
</tr>
<tr>
<td>• Source code: my.documents.lisp</td>
</tr>
</tbody>
</table>

- Documentation: Not supplied yet.
- Has-Author:
  - Maximum-Slot-Cardinality: 0 to 1
  - Maximum-Slot-Cardinality: 0 to 1
- Publisher:
  - Else-Cardinality: 0 to 1
- Title:
  - Else-Cardinality: 0 to 1

Description Logic

- There is a family of Frame-like KR systems with a formal semantics.
  - E.g., KL-ONE, LOOM, Classic, ...
- An additional kind of inference done by these systems is automatic **classification**
  - finding the right place in a hierarchy of objects for a new description
- Many current systems take care to keep the language simple, so that all inference can be done in polynomial time (in the number of objects)
  - ensuring tractability of inference
Emerging Paradigms (‘70’s - ‘80’s)

- Semantic nets
- Frames
- Production rule systems
  - Situation-action rules
    - If (warning light on) then (turn off engine)
  - If-then inference rules
    - If (warning light on) then (engine overheating)
    - If (warning light on) then ((engine overheating) 0.95)
- Hybrid procedural-declarative representation
- Basis for expert systems
- Predicate logic

Production Systems

- The notion of a “production system” was invented in 1943 by Post
- Used as the basis for many rule-based expert systems
- Used as a model of human cognition in psychology
- A production is a rule of the form:
  \[ C_1, C_2, \ldots, C_n \Rightarrow A_1 A_2 \ldots A_m \]
  
  - **Left hand side (LHS)**
  - **Right hand side (RHS)**
  
  Condition which must hold before the rule is applied
  Actions to be performed or conclusions to be drawn when the rule is applied
Basic Components

- **Rules:** Unordered set of user-defined "if-then" rules.
  - Form: \(\text{if } P_1 \land \ldots \land P_m \text{ then } A_1, \ldots, A_n\)
  - The \(P_i\)s are facts that determine conditions when rule is applicable.
  - Actions can add or delete facts from the Working Memory.

- **Working Memory:** A set of "facts" consisting of positive literals defining what's known to be true about the world.
  - Usually "flat tuples" like (age finin 45)

- **Inference Engine:** Procedure for inferring changes (additions and deletions) to Working Memory.
  - Typically forward chaining

Typical CLIPS Rule

```
(defrule determine-gas-level
  (working-state engine does-not-start)
  (rotation-state engine rotates)
  (not (repair ?))
  =>
  (if (not (yes-or-no-p "Gas in tank?")
          then (assert (repair "Add gas."))))

(defrule normal-engine-state
  (declare (salience 10))
  (working state engine normal)
  =>
  (assert (repair "No repair needed.")
          (assert (spark-state engine normal))
          (assert (charge-state battery charged))
          (assert (rotation-state engine rotates))

(defrule print-repair
  (declare (salience 10))
  (repair ?item)
  =>
  (printout t crlf crlf)
  (printout t "Suggested Repair: ")
  (printout t crlf crlf)
  (format t " %s%n%n%n" ?item))
```
Typical CLIPS facts

- Facts in most production systems are basically flat tuples
- A simple extension supported by many is to allow simple templates using “slot filler” pairs.
  - (deftemplate engine
    (slot horsepower)
    (slot displacement)
    (slot manufacturer)
    (slot year))
- Matching slots in a template is order insensitive, as in:
  - (engine (year 1998) (horsepower ?x))
  - (engine (horsepower 250) (displacement 409) (manufacturer ford))

Basic Procedure

While changes are made to Working Memory do:

- **Match** -- Construct the Conflict Set -- the set of all possible (rule, list-of-facts) pairs such that rule is one of the rules and list-of-facts is a subset of facts in WM that unify with the antecedent part (i.e., Left-hand side) of the given rule.
- **Conflict Resolution** -- Select one pair from the Conflict Set for execution.
- **Act** -- Execute the actions associated with the consequent part of the selected rule, after making the substitutions used during unification of the antecedent part with the list-of-facts.
Rete Algorithm

- The **Rete Algorithm** (Greek for “net”) is the most widely efficient algorithm for the implementation of production systems.
- Rete is the only algorithm for production systems whose efficiency is asymptotically independent of the number of rules.
- The basis for a whole generation of fast expert system shells: OPS5, ART, CLIPS and Jess.

Soar

- Soar is a production system developed initially at CMU and now used in many places.
- Soar stood for State, Operator And Result because all problem solving in Soar is regarded as a search through a problem space in which you apply an operator to a state to get a result.
- It's also a general cognitive architecture for developing systems that exhibit intelligent behavior.
- See [http://ai.eecs.umich.edu/soar/](http://ai.eecs.umich.edu/soar/)
- Example:

```
sp {hello
(state <s> ^type state)
    ^world
(write |Hello world|)
(halt)}
```
Emerging Paradigms ('70’s - '80’s)

- Semantic Nets
- Frames
- Production rule systems
- Predicate calculus
  - Primarily first order logic
    “Everybody loves somebody sometime.”
    $$(\forall p \ (\exists q \ (\exists r \ (p = q \land q = r \land p = r)))$$
- Resolution theorem proving

KR in the ‘90’s

- Declarative representations
  - Easier to change
  - Multi-use
  - Extendable by reasoning
  - Accessible for introspection
- Formal semantics
  - Defines what the representation means
  - Specifies correct reasoning
  - Allows comparison of representations/algorithms
- KR rooted in the study of logics
  - Temporal, context, modal, default, nonmonotonic, ...
- Rigorous theoretical analysis
**Description Logic**
- There is a family of Frame-like KR systems with a formal semantics.
- E.g., KL-ONE, LOOM, Classic, ...
- An additional kind of inference done by these systems is automatic **classification**
  - finding the right place in a hierarchy of objects for a new description
- Current systems take care to keep the language simple, so that all inference can be done in polynomial time (in the number of objects)
  - ensuring tractability of inference

**KR in the ‘00’s ??**
- Web based systems
  - embedding knowledge on web pages
  - languages based on XML: OIL, RDF, DAML, OWL
- Driven by new classes of applications
  - Electronic commerce (e.g., product catalogues)
  - Information retrieval on the web
- Integration with conventional software
  - e.g., OO modeling tools like UML
  - e.g., reflection in Java
- Business rules?
- Ontologies !
- ??
Summary

- Real knowledge representation and reasoning systems come in several major varieties.
- These differ in their intended use, degree of formal semantics, expressive power, practical considerations, features, limitations, etc.
- Some major families are
  - Logic programming languages
  - Theorem provers
  - Rule-based or production systems
  - Semantic networks
  - Frame-based representation languages
  - Databases (deductive, relational, object-oriented, etc.)
  - Constraint reasoning systems
  - Description logics