Knowledge Representation and Search

- If a computer only has the right knowledge, representation of that knowledge, and way of indexing that knowledge then it can be intelligent.

Physical Symbol Hypothesis

- Intelligent activity in either human or machine requires [Newell & Simon]:
  - Symbol patterns to represent significant aspects of a problem domain
  - Operations on these patterns to generate potential solutions to problems
  - Search to select a solution from among these possibilities

Production Systems

- A particularly important model of computation in AI

Definition

- A production system has:
  - A set of productions: Each production consists of a condition-action pair and defines a single chunk of problem-solving knowledge
  - Working memory: contains a description of the current state of the world in a reasoning process. The actions of production rules are specifically designed to alter the contents of working memory.
  - Recognize-act cycle

The recognize-act cycle

- Working memory is initialized
- The current state is maintained as a set of patterns in working memory
- The patterns are matched against the conditions of the productions
- Multiple productions may match (there may be conflicts), but only one is chosen to be activated or fired. This means that the action for that production is performed
- The process terminates when the contents of working memory do not match any rule conditions

Do you know?

- What production systems have you seen?
- Expert systems are often built as production systems that have the ability to backtrack in a search
Comments on Eliza?
- In emacs:
  - M-x doctor

Conflict Resolution
- May be “pick the first rule that matches”
- May have a heuristic and be very complex

- What kinds of conflict resolution can you think of?

Towers of Hanoi

- Most algorithms are recursive, but they don’t need to be

The Production System Algorithm
- We stipulate that D3 and D1 must always move clockwise, and that D2 must always move counter-clockwise.
- We always move the largest disc that can be moved in its stipulated direction.
- Assume that a disc can be sensed by sensory features B1, B2, and B3.
- Bi has value 1 if and only if disc Di is the largest disc that can be moved in its stipulated direction. Otherwise, it has a value of 0.

Knowledge Bases (KBs)
- Inference Engine
  - Contains domain-independent algorithms
- Knowledge Base
  - A set of sentences in a formal language
  - The declarative approach to building a system: tell it what it needs to know
  - Contains domain-specific content
- Agents can be viewed from a couple of different perspectives
  - The knowledge level: what they know
  - Implementation level: data structures in KB and algorithms that manipulate them
Representation Schemes

- Schemes should be [page 36 in Luger]:
  - Expressive: The scheme must be adequate to express all necessary information
  - Efficient: Support efficient execution of the resulting code
  - Natural: Provide a natural scheme for expressing the required knowledge

An example

- Task Description
  - To write a program that finds, for a given phone number, all possible encodings by words, and prints them. A phone number is an arbitrary(!) string of dashes -, slashes / and digits. The dashes and slashes will not be encoded. The words are taken from a dictionary which is given as an alphabetically sorted ASCII file (one word per line).
  - Participants: 14 programmers (ave. experience: ~ 7 yr)
  - Biggest experimental flaw: subjects self selected

Mapping

- The following mapping from letters to digits is given:

<table>
<thead>
<tr>
<th>Letters</th>
<th>Digits</th>
</tr>
</thead>
<tbody>
<tr>
<td>E J N Q R W X D S Y F T A M</td>
<td>0 1 2 3 4 5</td>
</tr>
<tr>
<td>C I V B K U L O P G H Z</td>
<td>6 7 8 9</td>
</tr>
</tbody>
</table>

The Results

- Using Lisp
  - Time (hr): 2 to 8.5, ave: 5
  - Lines of Code: 51 to 182
  - Run Time (median): 30 seconds
- Using C/C++
  - Time (hr): 3 to 25, ave: 11
  - Lines of Code: 107 to 614, ave: 277
- Using Java
  - Time (hr): 4 to 63, ave: 9
  - Lines of Code: 107 to 614, ave: 277

A Good Representational Scheme

- Inferential efficiency: It should facilitate efficiency in accessing specific knowledge for a particular problem
- Acquisitional adequacy: It should support the addition of new knowledge to the database as the program is running
Key Issues in Knowledge Representation

- Representing basic factual knowledge
  - At the lowest level, we must represent basic facts – e.g. “snoopy is a bird”.
  - We need some sort of system for recording such factual knowledge in a way that’s easy to access.

Why?

- We want agents to accomplish a task in the world
- Agents should be able to:
  - Represent states, actions, etc.
  - Incorporate new perceptions about the world
  - Update internal representations of the world
  - Deduce hidden properties of the world
  - Deduce appropriate actions

Pico’s World

- Pico is a depressed pirate
  - Perceptions: smoke, glitter, smell
  - Actions: left turn, right turn, forward, grab, release, shoot
  - Goals: get gold without entering fire or snoopy square

Snoopy

- Everybody should fear the snoopy bird!
- www.snoopybird.com

Example World

Historically

- The search to bottle reason
  - Aristotle (384-322 B.C.): codified styles of deductive reasoning
  - Gottfried Leibniz (1646-1716): dreamed of a calculus philosophicus
  - George Boole (in 1854): developed the foundations of propositional logic.
  - Gottlieb Frege (1879): developed predicate calculus
Types of Logic

<table>
<thead>
<tr>
<th>Language</th>
<th>What exists?</th>
<th>What states of knowledge?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Propositional logic</td>
<td>Facts</td>
<td>True/False/Unknown</td>
</tr>
<tr>
<td>First-order logic</td>
<td>Facts, objects, relations</td>
<td>True/False/Unknown</td>
</tr>
<tr>
<td>Temporal logic</td>
<td>Facts, objects, relations, times</td>
<td>True/False/Unknown</td>
</tr>
<tr>
<td>Probability theory</td>
<td>Facts</td>
<td>Degree of belief 0..1</td>
</tr>
<tr>
<td>Fuzzy logic</td>
<td>Degree of truth</td>
<td>Degree of belief 0..1</td>
</tr>
</tbody>
</table>

"Using words, phrases, and sentences, we can represent and reason about properties and relationships in the world." [Luger p. 47]

Entailment

- KB |= α
  - Knowledge Base KB entails sentence α if and only if α is true in all worlds where KB is true
  - Examples:
    - the KB containing “The mushroom is purple” and “The mushroom tastes good” entails “The mushroom is purple or the mushroom tastes good”
    - x+y=4 entails 4=x+y
  - One can think of the truth of α as being contained in truth of KB

A Good Representational Scheme

- Representational adequacy: Should be able to represent all knowledge needed
- Inferential adequacy
  - Soundness: something can be inferred only if it’s entailed by the knowledge base
  - Completeness: anything that’s entailed by the knowledge base can be inferred
  - Decidability: if a statement’s true/false, it should be possible to infer its truth/falseness in finite time

Models

- Formally structured worlds with respect to which truth can be evaluated
- m is a model of a sentence α if α is true in m
- M(α) is the set of all models of α
- Then KB |= α ⇔ M(KB) ⊆ M(α)

Example World

<table>
<thead>
<tr>
<th>Smell</th>
<th>Smoke</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Smoke</td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>
Inference

- KB ⊨ α = sentence α can be derived from the KB by procedure i
- Soundness: i is sound if whenever KB ⊨ α, it is also true that KB ⊨ α
- Completeness: i is complete if whenever KB ⊨ α, it is also true that KB ⊨ α
- Note: there is a sound and complete inference procedure for the logic we’re defining

Propositional Logic Semantics

- Each model specifies true/false for each proposition symbol
- Rules for evaluation truth with respect to a model m:
  - ¬S is true iff S is false
  - S₁ ∧ S₂ is true iff S₁ is true AND S₂ is true
  - S₁ ∨ S₂ is true iff S₁ is true OR S₂ is true
  - S₁ ⇒ S₂ is true iff S₁ is false OR S₂ is true
  - S₁ ⇔ S₂ is true iff S₁ ⇒ S₂ is true AND S₂ ⇒ S₁ is true

Reminder

- What are these English sentences in propositional logic?
  - There is a smell in 2,1 if and only if there is a snoopy in either 2,2 or 3,1.
  - There is no snoopy in 1,1
  - There is smoke in 2,1 if and only if there is a fire in either 2,2 or 3,1

Validity and Satisfiability

- A sentence is valid if it’s true in all models
- A sentence is satisfiable if it is true in some model
- A sentence is unsatisfiable if it is true in no models
- Satisfiability is connected to inference via the following:
  - KB ⊨ α if and only if (KB ∧ ¬α) is unsatisfiable

Normal Forms

- Other approaches to inference use syntactic operations on sentences, often expressed in standardized forms
  - Conjunctive Normal Form (CNF)
    - Conjunction of disjunctions of literals
      - (A ∨ ¬B) ∧ (B ∨ ¬C ∨ ¬D)
  - Disjunctive Normal Form (DNF)
    - Disjunction of conjunctions of literals
      - (A ∧ B) ∨ (A ∧ ¬C) ∨ (¬B ∧ ¬C) ∨ (A ∧ ¬D)
  - Horn Form
    - Conjunction of Horn clauses (clauses with ≤1 positive literal)

Proof Methods

- Model checking
  - Truth table enumerating (sound and complete for propositional logic)
  - Heuristic search in model space
- Application of inference rules
  - Legitimate (sound) generation of new sentences from old
  - Proof: a sequence of inference rule applications. We can use inference rules as operators in a standard search algorithm.
Inference Rules for Propositional Logic

- Resolution (for CNF): complete for propositional logic
  \[ \alpha \lor \beta, \neg \beta \lor \gamma \quad \frac{\alpha \lor \gamma}{\alpha \lor \gamma} \]

- Modus Ponens (for Horn Form): complete for Horn KBs
  \[ \alpha_1, \ldots, \alpha_n, \alpha_1 \land \ldots \land \alpha_n \Rightarrow \beta \quad \frac{\beta}{\beta} \]

Inference Rules

- Modus tollens:
  \[ P \Rightarrow Q \text{ is known to be true and } Q \text{ is known to be false, then we can infer } \neg P \]

- And elimination: \[ P \land Q \text{ is known to be true then we can conclude } P \text{ and } Q \]

- And introduction: vice-versa of above

- Universal instantiation:
  \[ \text{If } a \text{ if from the domain of } x, \quad \forall x \ P(x) \text{ lets us infer } P(a) \]

Resolution

- Ch. 7
- A complete inference algorithm when coupled with any complete search algorithm

Resolution: how

- Procedure:
  - Put premises/axioms into clause form
  - Add negation of what is to be proved, in clause form, to the set of axioms.
  - Resolve these clauses together, producing new clauses that logically follow from them.
  - Produce a contradiction by generating the empty clause.

- This is reductio ad absurdum (proof by reduction to an absurd thing) or proof by contradiction

Proving in Pico’s World

- If pico has visited [1,1], [2,1], and [1,2], prove that snoopy is in square [3,1]
  - What is in the KB?

Example World
Another problem

- Given the following, can you prove that the unicorn is mythical? Magical? horned?
  - If the unicorn is mythical, then is is immortal, but if it is not mythical, then it is a mortal mammal. If the unicorn is either immortal or a mammal, then it is horned. The unicorn is magical if it is horned.

For you to do

- If there is goodness or happiness then there are no jelly beans or killing. There is goodness. Prove: There are no jelly beans.
- If there are people then it is not quiet. There is no rest if it is not quiet. Prove: If there are people there is no rest.
- Prove: ((P ∨ Q) ∧ ¬P) → Q

Resolution Heuristics

- Clause elimination:
  - Certain types of clauses can be ignored
    - Pure clauses (where P appears and ¬P doesn’t appear elsewhere)
    - Tautologies
- Resolving:
  - Unit clauses first
  - Start with what you’re trying to prove
  - Aim to shorten clauses

Problems?

- What kinds of problems exist for propositional logic?

Predicate Calculus

- A way to access the components of an individual assertion
- Predicate Calculus: used extensively in many AI programs, especially expert systems

Syntax

- Constants: Jessica, 2, RIT
- Predicates: Sister, >, …
- Functions: Sqrt, Eat, …
- Variables: x, y, a, b
- Connectives: ∧ ∨ ¬ ⇒ ⇔
- Equality: =
- Quantifiers: ∀ ∃
Atomic Sentences

- Atomic sentences:
  - `predicate(term_1, ..., term_n)`
  - `term_1 = term_2`

- Term:
  - `function(term_1, ..., term_n)`
  - `or constant`
  - `or variable`

- Examples
  - `Daughter(Lisa, Homer)`
  - `Length(LeftLegOf(Homer)), Length(LeftLegOf(Marge)))`

Truth

- Sentences are true with respect to a model and an interpretation
- Models contain objects and relations among them
- Interpretation specifies values referred to for
  - `Constant symbols`
  - `Predicate symbols`
  - `Function symbols`

- An atomic sentence `predicate(term_1, ..., term_n)` is true if the objects referred to by `term_1, ..., term_n` are in the relation referred to by `predicate`

Complex Sentences

- Complex sentences are made from atomic sentences using connectives
  - `¬S` `S_1 ∧ S_2` `S_1 ∨ S_2` `S_1 → S_2` `S_1 ⇔ S_2`

- Examples:
  - `Sibling(Pico, HAL) ⇒ Sibling(HAL, Pico)`
  - `(42,3) ≤ (42,3)`
  - `(42,3) ∧ (42,3)`

Universal Quantification

∀ <variables> <sentence>

- “Everyone in Computer Science is smart”

  ∀x In(x, ComputerScience) ⇒ Smart(x)

- ∀x P is equivalent to the conjunction of instantiations of P

  In(Matt, ComputerScience) ⇒ Smart(Matt)
  ∧ In(Dan, ComputerScience) ⇒ Smart(Dan)
  ∧ In(Tim, ComputerScience) ⇒ Smart(Tim)
  ∧ ...

Existential Quantification

∃ <variables> <sentence>

- “Somebody in Computer Science is smart”

  ∃x In(x, ComputerScience) ∧ Smart(x)

- ∃x P is equivalent to the disjunction of instantiations of P

  In(Matt, ComputerScience) ∧ Smart(Matt)
  ∨ In(Dan, ComputerScience) ∧ Smart(Dan)
  ∨ In(Tim, ComputerScience) ∧ Smart(Tim)
  ∨ ...

Universal Quantification Common Mistake

- Typically, ⇒ is the main connective with ∀.
- Common mistake: using ∧ as the main connective

  ∀x In(x, ComputerScience) ∧ Smart(x)

- Means “Everyone is in Computer Science and everyone is smart”
### Existential Quantification

#### Common Mistake

- Typically, $\land$ is the main connective with $\exists$.
- Common mistake: using $\Rightarrow$ as the main connective
  
  $\exists x \text{ In}(x, \text{ComputerScience}) \Rightarrow \text{Smart}(x)$

- Is true if there’s anyone who’s not in Computer Science!

---

### More about Quantifiers

- $\forall \exists \forall$ is the same as $\exists \forall \forall$
- $\exists \forall \exists$ is the same as $\forall \exists \forall$
- $\exists \forall \forall$ is not the same as $\forall \exists \exists$

- There is a person who loves everyone in the world. $\exists x \forall y \text{ Loves}(x,y)$
- Everyone in the world is loved by at least one person $\forall x \exists y \text{ Loves}(x,y)$

---

### Quantifier Duality

- Each can be expressed using the other

  \[
  \forall x \text{ Eats}(x, \text{Mushrooms}) \equiv \neg \exists x \neg \text{Eats}(x, \text{Mushrooms})
  \]

  \[
  \exists x \text{ Eats}(x, \text{Vegetables}) \equiv \neg \forall x \neg \text{Eats}(x, \text{Vegetables})
  \]

---

### English into First Order Logic

- Can you do these?
  - Every dog hates cats.
  - All purple mushrooms are tasty.
  - You can fool some of the people all of the time.
  - You can fool all of the people some of the time.
  - [answers discussed in class]

---

### Equality

- $\text{term}_1 = \text{term}_2$ is true under a given interpretation if and only if $\text{term}_1$ and $\text{term}_2$ refer to the same object
  
  - $1=2$ and $\forall x \quad * (\text{Sqrt}(x), \text{Sqrt}(x)) = x$ are satisfiable
  - $2=2$ is valid

- Definition of (full) Sibling in terms of Parent
  
  \[
  \forall x, y \quad \text{Sibling}(x, y) \iff \neg (x = y) \land \exists m, f \quad \neg (m = f) \land \\
  \text{Parent}(m, x) \land \text{Parent}(f, x) \land \text{Parent}(m, y) \land \text{Parent}(f, y)
  \]