Knowledge Representation

• The representation used can influence the speed and efficiency of a system.
• There are two kinds of entities to be dealt with:
  – Facts or truths in some relevant world.
    • These are what we want to represent.
  – Representations of facts in some chosen formalism.
    • These are what will actually be manipulated.

Knowledge Representation

• The diagram shows how the facts and their representations are linked.
• The links are called representation mappings.
  – Forward representation maps from facts to representation.
  – Backward representation maps from representation to facts.

Knowledge Representation

• Have already considered one representation: English language.
  – Remember Spot and his tail.
  – Spot is a dog
    • represented by dog(Spot).
  – Also have fact that all dogs have tails
    • \( \forall x: \text{dog}(x) \rightarrow \text{hastail}(x) \).

Knowledge Representations

– Using deductive mechanisms we can show:
  • hastail(Spot)
– And with an appropriate backward mapping, we can produce
  • Spot has a tail.
• There is not always a one to one mapping.
  – More often there is a one to many or many to many relationship.
  – Consider the following:-

Knowledge Representation

• “All dogs have tails”
• “Every dog has a tail”
  – could both represent the same fact.
  – Or that all dogs have more than one tail.
  – Or that every dog has at least one tail.
• Representing English is a two part process
  – 1. What does the sentence actually mean.
  – 2. Convert those fact(s) into the new representation.
Knowledge Representation

- The above links are the key components for any KB program.
  - An AI program manipulates the internal representations of the facts it is given.
  - This should result in new structures that can also be interpreted.
    - The new structures should be the internal representation of the facts of the answer.

Facts \(\rightarrow\) Internal Representations

A good representation makes the operation of a reasoning program trivial.
- Consider the mutilated chessboard.
- The representations is important because:
  - when backward mapping is applied to the program output, the appropriate final facts will be generated.

- If either the program operation or one of the representations of facts are not faithful to the problem, the result will not be correct.
- If no good mapping for a problem can be defined, then no matter how good the program, it will not be able to produce answers corresponding to real answers to the problem.

- Four properties for any good system representing knowledge.
  1. Representational Adequacy
     - the ability to represent all of the kinds of knowledge that are needed in that domain.
  2. Inferential Adequacy
     - the ability to manipulate the representational structures in such a way as to derive new structures corresponding to new knowledge inferred from old.
  3. Inferential Efficiency
     - the ability to incorporate into the knowledge structure additional information that can be used to focus the attention of the inference mechanisms in the most promising directions.
  4. Acquisitional Efficiency
     - the ability to acquire new information easily. The simplest case involves direct insertion, by a person, of new knowledge into the database. Ideally, the program itself would be able to control knowledge acquisition.

No systems as yet exists that optimises all capabilities for all kinds of knowledge.
- Have already looked at some ways to represent knowledge
  - predicate logic.
- Look now at some other issues in knowledge and its representation.
Knowledge Representation

• Simple Relational Knowledge
  – simplest way to represent knowledge is as a set of relations.
  • Similar to creating a database table.
  – It may be input to powerful inference engines to be manipulated.
  • On its own, it is not much good, but using it to answer questions becomes a relatively simple matter.

Knowledge Representation

• Inheritable Knowledge
  – knowledge about objects, their attributes and their values need not be as simple as a DB table.
  – It is possible to augment the basic representation with inference mechanisms that operate on the structure.
  – The representation must be defined with this in mind.

Knowledge Representation

– A useful form is property inheritance.
  • Elements of specific classes inherit attributes and values from more general classes in which they are included.
  – Requires the organisation of objects into classes and the formation of a class hierarchy.
  – In the diagram on the next slide
    • lines represent attributes and boxed nodes represent objects and values of attributes of objects.

Knowledge Representation

• Important Relations
  – ISA which is used to show class inclusion.
  – Instance which is used to show class membership.
  • Provide the basis for property inheritance as an inference technique.
  – Allow the retrieval both of facts explicitly stored and those that can be derived from those explicitly stored.

Knowledge Representation

• A simplified algorithm to retrieve a value V for attribute A of an instance object O:-
  • find O in the KB
  • if there is a value for A, report that value
  • otherwise, is there a value for the attribute instance, if not fail
  • otherwise move to the node corresponding to the value and look for a value for A, report if found.
  • Otherwise
Knowledge Representation

- Otherwise, do until there is no value for the isa attribute or until an answer is found
  - 1. Get the value of the isa attribute and move to that node
  - 2. See if there is a value for the attribute A, report if found.
- Simplistic procedure, but it does illustrate the point.

Knowledge Representation

- Inferential Knowledge
  - have seen this with predicate logic.
  - The inference mechanism implements the standard logical rules of inference.
    - Can either reason forward from facts, or backwards from conclusions - it depends.
    - Often useful to combine logic with with an isa hierarchy to provide a more powerful system
  - Logic is a good way to describe the relations between attributes.

Knowledge Representation

- Procedural Knowledge
  - operational knowledge that specifies what to do when…
  - most common way is to represent it as code (LISP for example)
  - The machine uses the knowledge when it executes the code

Knowledge Representation

- Presents difficulties in the area of inferential adequacy
  - how do you write a program to reason about the behaviour of another program.
- Also, for acquisitional efficiency
  - updating and debugging large pieces of code is not easy.
- Because of these:-

Knowledge Representation

- better ways of representing procedural knowledge have been developed.
  - Production rules, rules about rules
- But there are issues here as well
  - declarative and procedural meaning of code.

Knowledge Representation

- Issues:-
  - are any attributes of objects so basic that they occur in almost every problem domain?
  - Are there important relationships among attributes of objects?
  - At what level should knowledge be represented? What should the primitives be?
  - How should objects be represented?
  - Given a large amount of data, how do we get the right parts?
Knowledge Representation

• Important Attributes
  – isa and instance
  – Designate class membership and class inclusion.
  – Can be represented explicitly or by rules describing particular classes.

• Relationships among Attributes
  – what properties do attributes have independent of the specific knowledge they encode?

  – Four to consider.
  – 1. Inverses
    • describing a relationship as an attribute, we commit to a perspective. We focus on one object and look for binary relations between it and others.
    • Seen this with instance, isa and team in the baseball example.
      – Each shown with a directional arrow.
      – No matter which way the arrow goes there will be a relationship.
      – It can be important to represent both perspectives.

  – 2. An ISA hierarchy of attributes
    • in the same way as hierarchy of classes
    • height is a specialisation of the more general physical-size which is a specialisation of physical attribute.
    • They support inheritance about constraints on values and the mechanisms for computing the values.

  – 3. Techniques for reasoning about values
    • some values are specified explicitly, but the system may have to reason about others.

  – 4. Single Value Attributes
    • takes a unique value.
    • If the value is present and a difference value is asserted, then one of two things has happened.
Knowledge Representation

- The Granularity of Representation
  - at what level of detail should the world be represented?
  - What should be the primitives?
  - How do you represent “John spotted Sue”?
    - spotted(John, Sue) or saw(John, Sue)
  - The major advantage
    - rules that are used to derive inferences from that knowledge need be written in terms of small primitives.

- Really an argument for a simple canonical form.
  - There are arguments for and against using low level primitives
    - Simple high level facts may require a lot of storage when broken into primitives.
    - Redundant storing of information for higher level actions.
    - John punched Mary and Mary punched John.
  - It may not be clear what the primitives should be in a particular domain.

- Representing Sets of Objects
  - Important to represent sets of objects.
  - Can be more efficient to associate a property once with a set than to explicitly associate it with each individual element.
  - Consider the assertion that “English speakers can be found all over the world”
    - Must attach the assertion to the set of English speakers as no single speaker can be found all over the world.

- Two ways to state a definition for a set and its elements.
  - 1. List the members. The extensional definition.
  - 2. Provide a rule that, when a particular object is evaluated, returns T/F indicating membership. The intensional definition.
    - Example:-
      - the set of our sun’s planets on which people live.
Knowledge Representation

- Extensional: \( \{ \text{Earth} \} \)
- Intensional: \( \{ x : \text{sun-planet}(x) \land \text{human-inhabited}(x) \} \)
- There is not a 1:1 relation between these types of definition. For every extensional definition, there may be more than one intensional definition.
- Determining equality of sets is more difficult with intensional definitions.

Knowledge Representation

- Intensional Definitions have some advantages:
  - can describe infinite sets and sets whose elements are not all known: prime numbers and kings of England.
  - They can describe the set depending on parameters that may change over time: "the president of the US was once a democrat".

Knowledge Representation

- Finding the Right Structures as Needed
  - how to locate the appropriate knowledge in memory.
  - Take the following: 
  - John went to Steak&Ale last night. He ordered a large rare steak, paid his bill and left.
  - The question is did John eat dinner last night?
  - Eating was not mentioned. But if one goes to a restaurant, eating is in the script for this.

Knowledge Representation

- So to solve a problem, the system will have scripts for every situation - from sailing around the world to going to work.
  - But the description didn’t mention “restaurant” either.
  - So how do we select the appropriate script?
  - Minsky suggests that there are a number of problems in selecting.
  - These are:

Knowledge Representation

- How to perform an initial selection of the most appropriate structure?
- How to fill in appropriate details from the current situation.
- How to find a better structure if the one chosen initially turns out not to be appropriate?

Knowledge Representation

- What to do if none of the available structures is appropriate?
- When to create and remember a new structure?
Knowledge Representation

• Selecting an initial structure
  – Several approaches.
  – Index the structures by significant English words.
    • Difficult approach in practise, but not unusable, words have many different meanings in different contexts, so these have to be accounted for.

Knowledge Representation

– Consider each major concept as a pointer to all of the potential scripts.
  • Back to the steak. Steak may produce the concept of supermarket or restaurant.
  • Take all the clues and try to reason which is the correct script to invoke.
– Locate one major clue in the description and use it to select an initial structure.
  • As other clues appear, use them to refine the choice of structure.

Knowledge Representation

– None of these options is foolproof and with larger structures, it is often harder to tell which is appropriate.
• Revising the choice when necessary
  – with the first candidate structure to hand, a detailed match must be made.
    • Have seen an example of this in logic and substitution.
  – If no appropriate match can be found, a new Structure must be selected.

Knowledge Representation

– The way we chose the first structure may help choose the second.
  – Select fragments of the current structure that match the problem, use these to find another structure.
  – Make an excuse for the shortcomings of the first and continue to use it.
    • Something may be broken.
  – Refer to specific stored links between structures.

Knowledge Representation

– If the knowledge structures are stored in an isa hierarchy, the traverse upwards until the structure found is sufficiently general that it does not conflict.

Knowledge Representation

• The Frame Problem
  – A problem which arises is how to represent efficiently sequences of problem states that arise from a search process.
  – Consider a household robot.
    • Many objects and relations in the world and a state description must include facts like
      – on(Plant12, Table34).
    • One strategy is to store each state description as a list of such facts.
Knowledge Representation

• However, most of these facts will not change, but will still be represented at each node - run out of memory.
• A lot of time will be spent copying these nodes.
• And there is the additional problem of figuring out which facts should be different at each node.
• The problem of representing facts that do change as well as those that don’t is called The Frame Problem.

Knowledge Representation

• Some systems use frame axioms for reasoning.
  – Frame axioms describe all the things that do not change when an operator is applied in state n.
  – All things that do change are mentioned as part of the operator itself.
  – In any large domain, this becomes unwieldy as the number of axioms grows.

Knowledge Representation

• Could start with the initial state and apply the changes as given by the rules applied.
• This solves the memory problem and is good until backtracking occurs.
  – Unless all changes can be ignored, there is a problem.
  – So what changes need to be undone?
  – Two ways to solve this problem.

Knowledge Representation

• 1. Do not modify the initial state description.
  – At each node, store an indication of the changes.
  – When a current description is required, build it as necessary.
  – Backtracking is easy, but referring to the current state is now made much more complex.

Knowledge Representation

• 2. Modify the initial state.
  – Store an indication of what to do to undo the move in the case of backtracking.
  – During backtracking, perform these reverse operations as allowed.
  – Sometimes these solutions are not enough.

Knowledge Representation

• may want to remember that before the table was moved, it was under the window and after it was moved it is in the centre of the room.
• Add to the representation of each fact an indication of the time at which it was true.
  – This is a state or situation variable.
• Still difficult in the real world.
Knowledge Representation

- Situations
  - Logical terms consisting of the initial situations (S₀) and all the situations that are generated by applying an action to a situation.
- The function result(a, s) names the situation that results when action a is executed in situation S. Consider positions when going through a maze.

Knowledge Representation

- The variable S₀ is a situation variable such that:
  - At(hunter, 1, 1) becomes at(hunter, 1, 1, S₀) which can be stated as at(hunter, 1, 1) is true in situation S₀.