UNIT-3

KNOWLEDGE REPRESENTATION

KNOWLEDGE REPRESENTATION:-

For the purpose of solving complex problems c\encountered in AI, we need both a large amount of knowledge and some mechanism for manipulating that knowledge to create solutions to new problems. A variety of ways of representing knowledge (facts) have been exploited in AI programs. In all variety of knowledge representations , we deal with two kinds of entities.

A. Facts: Truths in some relevant world. These are the things we want to represent.

B. Representations of facts in some chosen formalism . these are things we will

actually be able to manipulate.

One way to think of structuring these entities is at two levels : (a) the knowledge level, at which facts are described, and (b) the symbol level, at which representations of objects at the knowledge level are defined in terms of symbols that can be manipulated by programs.

The facts and representations are linked with two-way mappings. This link is called representation mappings. The forward representation mapping maps from facts to representations. The backward representation mapping goes the other way, from representations to facts.

One common representation is natural language (particularly English) sentences. Regardless of the representation for facts we use in a program , we may also need to be concerned with an English representation of those facts in order to facilitate getting information into and out of the system. We need mapping functions from English sentences to the representation we actually use and from it back to sentences.

Representations and Mappings

- In order to solve complex problems encountered in artificial intelligence, one needs both a large amount of knowledge and some mechanism for manipulating that knowledge to create solutions.
- Knowledge and Representation are two distinct entities. They play central but distinguishable roles in the intelligent system.
- Knowledge is a description of the world. It determines a system’s competence by what it knows.
- Moreover, Representation is the way knowledge is encoded. It defines a system’s performance in doing something.
- Different types of knowledge require different kinds of representation.

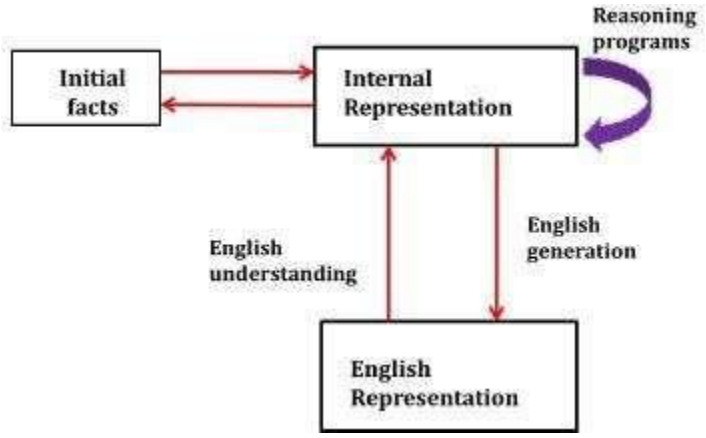


Fig: Mapping between Facts and Representations

The Knowledge Representation models/mechanisms are often based on:

- Logic
- Rules
- Frames
- Semantic Net

Knowledge is categorized into two major types:

1. Tacit corresponds to “informal” or “implicit“
    - Exists within a human being;
    - It is embodied.
    - Difficult to articulate formally.
    - Difficult to communicate or share.
    - Moreover, Hard to steal or copy.
    - Drawn from experience, action, subjective insight
2. Explicit formal type of knowledge, Explicit
    - Explicit knowledge
    - Exists outside a human being;
    - It is embedded.
    - Can be articulated formally.
    - Also, Can be shared, copied, processed and stored.
    - So, Easy to steal or copy
    - Drawn from the artifact of some type as a principle, procedure, process, concepts.

A variety of ways of representing knowledge have been exploited in AI programs.

There are two different kinds of entities, we are dealing with.

1. Facts: Truth in some relevant world. Things we want to represent.
2. Also, Representation of facts in some chosen formalism. Things we will actually be able to manipulate.

These entities structured at two levels:

1. The knowledge level, at which facts described.
2. Moreover, The symbol level, at which representation of objects defined in terms of symbols that can manipulate by programs

USING PREDICATE LOGIC

Representation of Simple Facts in Logic

Propositional logic is useful because it is simple to deal with and a decision procedure for it exists.

Also, In order to draw conclusions, facts are represented in a more convenient way as,

1. Marcus is a man.
    - man(Marcus)
2. Plato is a man.
    - man(Plato)
3. All men are mortal.
    - mortal(men)

But propositional logic fails to capture the relationship between an individual being a man and that individual being a mortal.

- How can these sentences be represented so that we can infer the third sentence from the first two?
- Also, Propositional logic commits only to the existence of facts that may or may not be the case in the world being represented.
- Moreover, It has a simple syntax and simple semantics. It suffices to illustrate the process of inference.
- Propositional logic quickly becomes impractical, even for very small worlds.

Predicate logic

First-order Predicate logic (FOPL) models the world in terms of

- Objects, which are things with individual identities
- Properties of objects that distinguish them from other objects
- Relations that hold among sets of objects

- Functions, which are a subset of relations where there is only one "value" for any given "input"

First-order Predicate logic (FOPL) provides

- Constants: a, b, dog33. Name a specific object.
- Variables: X, Y. Refer to an object without naming it.
- Functions: Mapping from objects to objects.
- Terms: Refer to objects
- Atomic Sentences: in(dad-of(X), food6) Can be true or false, Correspond to propositional symbols P, Q.

A well-formed formula (*wff*) is a sentence containing no "free" variables. So, That is, all variables are "bound" by universal or existential quantifiers.

(∀x)P(x, y) has x bound as a universally quantified variable, but y is free.

Quantifiers

Universal quantification

- (∀x)P(x) means that P holds for all values of x in the domain associated with that variable
- E.g., (∀x) dolphin(x) → mammal(x)

Existential quantification

- (∃ x)P(x) means that P holds for some value of x in the domain associated with that variable
- E.g., (∃ x) mammal(x) ∧ lays-eggs(x)

Also, Consider the following example that shows the use of predicate logic as a way of representing knowledge.

1. Marcus was a man.
2. Marcus was a Pompeian.
3. All Pompeians were Romans.
4. Caesar was a ruler.
5. Also, All Pompeians were either loyal to Caesar or hated him.
6. Everyone is loyal to someone.
7. People only try to assassinate rulers they are not loyal to.
8. Marcus tried to assassinate Caesar.

The facts described by these sentences can be represented as a set of well-formed formulas (*wffs*) as follows:

1. Marcus was a man.
    - man(Marcus)
2. Marcus was a Pompeian.
    - Pompeian(Marcus)
3. All Pompeians were Romans.
    - ∀x: Pompeian(x) → Roman(x)
4. Caesar was a ruler.
    - ruler(Caesar)
5. All Pompeians were either loyal to Caesar or hated him.
    - inclusive-or
    - ∀x: Roman(x) → loyalto(x, Caesar) ∨ hate(x, Caesar)
    - exclusive-or
    - ∀x: Roman(x) → (loyalto(x, Caesar) ∧¬ hate(x, Caesar)) ∨
    - (¬loyalto(x, Caesar) ∧ hate(x, Caesar))

6. Everyone is loyal to someone.
    - ∀x: ∃y: loyalto(x, y)
7. People only try to assassinate rulers they are not loyal to.
    - ∀x: ∀y: person(x) ∧ ruler(y) ∧ tryassassinate(x, y)
    - →¬loyalto(x, y)
8. Marcus tried to assassinate Caesar.
    - tryassassinate(Marcus, Caesar)

Now suppose if we want to use these statements to answer the question: ***Was Marcus loyal to Caesar?***

Also, Now let's try to produce a formal proof, reasoning backward from the desired goal: ¬ Ioyalto(Marcus, Caesar)
In order to prove the goal, we need to use the rules of inference to transform it into another goal (or possibly a set of goals) that can, in turn, transformed, and so on, until there are no unsatisfied goals remaining.

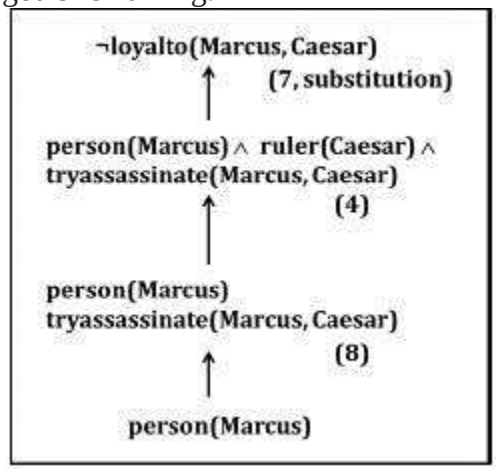


Figure: An attempt to prove ¬loyalto(Marcus, Caesar).

- The problem is that, although we know that Marcus was a man, we do not have any way to conclude from that that Marcus was a person. Also, We need to add the representation of another fact to our system, namely***: ∀man(x) → person(x)***
- Now we can satisfy the last goal and produce a proof that Marcus was not loyal to Caesar.
- Moreover, From this simple example, we see that three important issues must be addressed in the process of converting English sentences into logical statements and then using those statements to deduce new ones:
    1. Many English sentences are ambiguous (for example, 5, 6, and 7 above). Choosing the correct interpretation may be difficult.
    2. Also, There is often a choice of how to represent the knowledge. Simple representations are desirable, but they may exclude certain kinds of reasoning.
    3. Similalry, Even in very simple situations, a set of sentences is unlikely to contain all the information necessary to reason about the topic at hand. In order to be able to use a set of statements effectively. Moreover, It is usually necessary to have access to another set of statements that represent facts that people consider too obvious to mention.

Resolution Procedure

- Resolution is a procedure, which gains its efficiency from the fact that it operates on statements that have been converted to a very convenient standard form.
- Resolution produces proofs by refutation.
- In other words, ***to prove a statement (i.e., to show that it is valid), resolution attempts to show that the negation of the statement produces a contradiction with the known statements (i.e., that it is unsatisfiable).***
- The resolution procedure is a simple iterative process: at each step, two clauses, called the parent clauses, are compared (resolved), resulting in a new clause that has inferred from them. The new clause represents ways that the two parent clauses interact with each other. Suppose that there are two clauses in the system:

winter **V** ***summer***

¬ winter **V** ***cold***

- Now we observe that precisely one of winter and ¬ winter will be true at any point.
- If winter is true, then cold must be true to guarantee the truth of the second clause. If ¬ winter is true, then summer must be true to guarantee the truth of the first clause.
- Thus we see that from these two clauses we can deduce ***summer V cold***
- This is the deduction that the resolution procedure will make.
- Resolution operates by taking two clauses that each contains the same literal, in this example, ***winter***.
- Moreover, The literal must occur in the positive form in one clause and in negative form in the other. The resolvent obtained by combining all of the literals of the two parent clauses except the ones that cancel.
- If the clause that produced is the empty clause, then a contradiction has found.

For example, the two

c
l
a
u
s
e
s

w
i
n
t
e
r

¬

w
i
n
t
e
r
will produce the empty clause.

Procedural versus Declarative Knowledge

We have discussed various search techniques in previous units. Now we would consider a set of rules that represent,

1. Knowledge about relationships in the world and
2. Knowledge about how to solve the problem using the content of the rules.

Procedural vs
Declarative

Knowledge
Procedural
Knowledge

- A representation in which the control information that is necessary to use the knowledge is embedded in the knowledge itself for e.g. computer programs, directions, and recipes; these indicate specific use or implementation;
- The real difference between declarative and procedural views of knowledge lies in where control information reside.

For example, consider the following

M
a
n

(
M
a
r
c
u
s
)

M
a
n

(
C
a
e
s
a
r
)

P
e
r
s
o
n

(
C
l
e
o
p
a
t
r
a
)

∀x: Man(x) → Person(x)

Now, try to answer the question. ?Person(y)

The knowledge base justifies any of the following answers.
Y
=
M
a
r
c
u
s

Y
=
C
a
e
s
a
r

Y
=
C
l
e
o
p
a

t

r

a

- We get more than one value that satisfies the predicate.
- If only one value needed, then the answer to the question will depend on the order in which the assertions examined during the search for a response.
- If the assertions declarative then they do not themselves say anything about how they will be examined. In case of procedural representation, they say how they will examine.

Declarative Knowledge

- A statement in which knowledge specified, but the use to which that knowledge is to be put is not given.
- For example, laws, people's name; these are the facts which can stand alone, not dependent on other knowledge;
- So to use declarative representation, we must have a program that explains what is to do with the knowledge and how.
- For example, a set of logical assertions can combine with a resolution theorem prover to give a complete program for solving problems but in some cases, the logical assertions can view as a program rather than data to a program.
- Hence the implication statements define the legitimate reasoning paths and automatic assertions provide the starting points of those paths.
- These paths define the execution paths which is similar to the 'if then else "in traditional programming.
- So logical assertions can view as a procedural representation of knowledge.

Forward versus Backward Reasoning

Forward versus Backward Reasoning
A search procedure must find a path between initial and goal states. There are two directions in which a search process could proceed. The two types of search are:

1. Forward search which starts from the start state
2. Backward search that starts from the goal state

The production system views the forward and backward as symmetric processes. Consider a game of playing 8 puzzles. The rules defined are

Square 1 empty and square 2 contains tile n. →

- *Also, Square 2 empty and square 1 contains the tile n. Square 1 empty Square 4 contains tile n. →*
- *Also, Square 4 empty and Square 1 contains tile n.*

We can solve the problem in 2 ways:
1. Reason forward from the initial state

- Step 1. Begin building a tree of move sequences by starting with the initial configuration at the root of the tree.
- Step 2. Generate the next level of the tree by finding all rules ***whose left-hand side matches*** against the root node. The right-hand side is used to create new configurations.
- Step 3. Generate the next level by considering the nodes in the previous level and applying it to all rules whose left-hand side match.

2. Reasoning backward from the goal states:
   - Step 1. Begin building a tree of move sequences by starting with the goal node configuration at the root of the tree.
   - Step 2. Generate the next level of the tree by finding all rules ***whose right-hand side matches*** against the root node. The left-hand side used to create new configurations.
   - Step 3. Generate the next level by considering the nodes in the previous level and applying it to all rules whose right-hand side match.
   - So, The same rules can use in both cases.
   - Also, In forwarding reasoning, the left-hand sides of the rules matched against the current state and right sides used to generate the new state.
   - Moreover, In backward reasoning, the right-hand sides of the rules matched against the current state and left sides are used to generate the new state.

There are four factors influencing the type of reasoning. They are,

1. Are there more possible start or goal state? We move from smaller set of sets to the length.
2. In what direction is the branching factor greater? We proceed in the direction with the lower branching factor.
3. Will the program be asked to justify its reasoning process to a user? If, so then it is selected since it is very close to the way in which the user thinks.
4. What kind of event is going to trigger a problem-solving episode? If it is the arrival of a new factor, the forward reasoning makes sense. If it is a query to which a response is desired, backward reasoning is more natural.

Example 1 of Forward versus Backward Reasoning

- It is easier to drive from an unfamiliar place from home, rather than from home to an unfamiliar place. Also, If you consider a home as starting place an unfamiliar place as a goal then we have to backtrack from unfamiliar place to home.

Example 2 of Forward versus Backward Reasoning

- Consider a problem of symbolic integration. Moreover, The problem space is a set of formulas, which contains integral expressions. Here START is equal to the given formula with some integrals. GOAL is equivalent to the expression of the formula without any integral. Here we start from the formula with some integrals and proceed to an integral free expression rather than starting from an integral free expression.

Example 3 of Forward versus Backward Reasoning

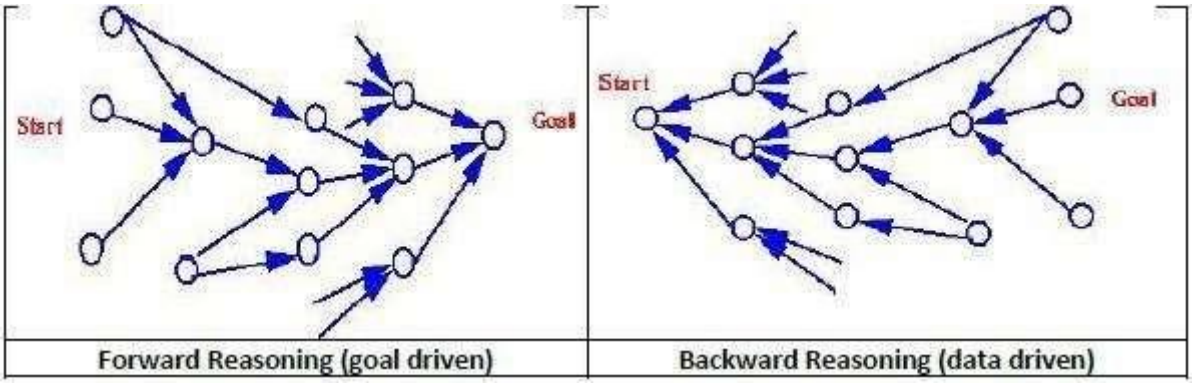
The third factor is nothing but deciding whether the reasoning process can justify its reasoning. If it justifies then it can apply. For example, doctors are usually unwilling to accept any advice from diagnostics process because it cannot explain its reasoning.

Example 4 of Forward versus Backward Reasoning

- Prolog is an example of backward chaining rule system. In Prolog rules restricted to Horn clauses. This allows for rapid indexing because all the rules for deducing a given fact share the same rule head. Rules matched with unification procedure. Unification tries to find a set of bindings for variables to equate a sub-goal with the head of some rule. Rules in the Prolog program matched in the order in which they appear.

Combining Forward and Backward Reasoning

- Instead of searching either forward or backward, you can search both simultaneously.
- Also, That is, start forward from a starting state and backward from a goal state simultaneously until the paths meet.
- This strategy called Bi-directional search. The following figure shows the reason for a Bidirectional search to be ineffective.



Forward versus Backward Reasoning

- Also, The two searches may pass each other resulting in more work.
- Based on the form of the rules one can decide whether the same rules can apply to both forward and backward reasoning.
- Moreover, If left-hand side and right of the rule contain pure assertions then the rule can reverse.
- And so the same rule can apply to both types of reasoning.
- If the right side of the rule contains an arbitrary procedure then the rule cannot reverse.
- So, In this case, while writing the rule the commitment to a direction of reasoning must make.