Representing Knowledge Using Rules

INT404

Procedural v/s Declarative Knowledge

- **Procedural v/s Declarative Knowledge**
  - A Declarative representation is one in which knowledge is specified but the use to which that knowledge is to be put in, is not given.
  - A Procedural representation is one in which the control information that is necessary to use the knowledge is considered to be embedded in the knowledge itself.
  - To use a procedural representation, we need to augment it with an interpreter that follows the instructions given in the knowledge.
  - The difference between the declarative and the procedural views of knowledge lies in where control information resides.

- **Procedural v/s Declarative Knowledge**
  - Consider the example
    - man(Marcus)
    - man (Ceaser)
    - Person(Cleopatra)
    - Vx : man(x) → person(x)
  - Now we want to extract from this knowledge base the ans to the question :
    - ∃y : person (y)
    - Marcus, Ceaser and Cleopatra can be the answers

✓Procedural v/s Declarative Knowledge

- ✓As there is more than one value that satisfies the predicate, but only one value is needed, the answer depends on the order in which the assertions are examined during the search of a response.
- ✓If we view the assertions as declarative, then we cannot depict how they will be examined. If we view them as procedural, then they do.
- ✓Let us view these assertions as a non deterministic program whose output is simply not defined, now this means that there is no difference between Procedural & Declarative Statements. But most of the machines don't do so, they hold on to what ever method they have, either sequential or in parallel.
- ✓The focus is on working on the control model.

✓**Procedural v/s Declarative Knowledge**

- ✓man(Marcus)
- ✓man (Ceaser)
- ✓Vx : man(x) → person(x)
- ✓Person(Cleopatra)

✓If we view this as declarative then there is no difference with the previous statement. But viewed procedurally, and using the control model, we used to got Cleopatra as the answer, now the answer is marcus.

✓The answer can vary by changing the way the interpreter works.

✓The distinction between the two forms is often very fuzzy. Rather then trying to prove which technique is better, what we should do is to figure out what the ways in which rule formalisms and interpreters can be combined to solve problems.

Logic Programming

✓Logic programming is a programming language paradigm in which logical assertions are viewed as programs, e.g : PROLOG

✓A PROLOG program is described as a series of logical assertions, each of which is a Horn Clause.

✓A Horn Clause is a clause that has at most one positive literal.

✓Eg p, ¬ p V q etc are also Horn Clauses.

✓The fact that PROLOG programs are composed only of Horn Clauses and not of arbitrary logical expressions has two important consequences.

- ✓Because of uniform representation a simple & effective interpreter can be written.
- ✓The logic of Horn Clause systems is decidable.

A Declarative and a Procedural Representation

$\forall x : pet(x) \bigwedge small(x) \rightarrow apartmentpet(x)$

$\forall x : cat(x) \bigvee dog(x) \rightarrow pet(x)$

$\forall x : poodle(x) \rightarrow dog(x) \bigwedge small(x)$

$poodle(ftujfy)$

A Representation in Logic

```
apartmentpet(X) :- pet(X), small(X).
pet(X) :- cat(X).
pet(X) :- dog(X).
dog(X) :- poodle (X) .
small(X) :- poodle(X).
poodle(fluffy).
```

A Representation in PROLOG

Answering Questions in PROLOG

```
apartmentpet(X) :- pet(X), small(X).
pet(X) :- cat(X).
pet(X) :- dog(X).
dog(X) :- poodle (X) .
small(X) :- poodle(X).
poodle(fluffy).

?- apartmentpet(X).

?- cat(fluffy).
```

Logic Programming

✓**Logic Programming**

✓Even PROLOG works on backward reasoning.

✓The program is read top to bottom, left to right and search is performed depth-first with backtracking.

✓There are some syntactic difference between the logic and the PROLOG representations as mentioned in Fig 6.1

✓The key difference between the logic & PROLOG representation is that PROLOG interpreter has a fixed control strategy, so assertions in the PROLOG program define a particular search path to answer any question.

Where as Logical assertions define set of answers that they justify, there can be more than one answers, it can be forward or backward tracking .

Logic Programming

- **Logic Programming**
  - Control Strategy for PROLOG states that we begin with a problem statement, which is viewed as a goal to be proved.
  - Look for the assertions that can prove the goal.
  - To decide whether a fact or a rule can be applied to the current problem, invoke a standard unification procedure.
  - Reason backward from that goal until a path is found that terminates with assertions in the program.
  - Consider paths using a depth-first search strategy and use backtracking.
  - Propagate to the answer by satisfying the conditions.

Forward v/s Backward Reasoning

✓**Forward v/s Backward Reasoning**

✓The objective of any search is to find a path through a problem space from the initial to the final one.

✓There are 2 directions to go and find the answer

✓Forward

✓Backward

✓8-square problem

✓**Reason forward from the initial states** : Begin building a tree of move sequences that might be solution by starting with the initial configuration(s) at the root of the tree. Generate the next level of tree by finding all the rules whose left sides match the root node and use the right sides to create the new configurations. Generate each node by taking each node generated at the previous level and applying to it all of the rules whose left sides match it. Continue.

Forward v/s Backward Reasoning

✓**Reason backward from the goal states** : Reason backward from the goal states : Begin building a tree of move sequences that might be solution by starting with the goal configuration(s) at the root of the tree. Generate the next level of tree by finding all the rules whose right sides match the root node and use the left sides to create the new configurations. Generate each node by taking each node generated at the previous level and applying to it all of the rules whose right sides match it. Continue. This is also called Goal-Directed Reasoning.

✓To summarize, to reason forward, the left sides(pre conditions) are matched against the current state and the right sides(the results) are used to generate new nodes until the goal is reached.

✓To reason backwards, the right sides are matched against the current node and the left sides are used to generate new nodes.

A Sample of the Rules for Solving the 8-Puzzle

Assume the areas of the tray are numbered:

1	2	3
4	5	6
7	8	9

Square 1 empty and Square 2 contains tile $n \rightarrow$
Square 2 empty and Square 1 contains tile n
Square 1 empty and Square 4 contains tile $n \rightarrow$
Square 4 empty and Square 1 contains tile n
Square 2 empty and Square 1 contains tile $n \rightarrow$
Square 1 empty and Square 2 contains tile n

⋮

An Examples :

Start

2	8	3
1	6	4
7		5

Goal

1	2	3
8		4
7	6	5

✓Forward v/s Backward Reasoning

✓Factors that influence whether to choose forward or backward reasoning :

✓Are there **more possible start states or goal states**? We would like to go from smaller set of states to larger set of states.

✓In which direction is the **branching factor** (the average number of nodes that can be reached directly from a single node) greater? We would like to proceed in the direction with the lower branching factor.

✓Will the program be **asked to justify its reasoning process** to the user? It so, it is important to proceed in the direction that corresponds more closely with the way user will think.

✓What kind of **event is going to trigger a problem-solving episode**? If it is the arrival of a new fact , forward reasoning should be used. If it a query to which response is desired, use backward reasoning.

Forward v/s Backward Reasoning

- **Forward v/s Backward Reasoning**
  - Home to unknown place example.
  - MYCIN
  - Bidirectional Search (The two searches must pass each other)
  - Forward Rules : which encode knowledge about how to respond to certain input configurations.
  - Backward Rules : which encode knowledge about how to achieve particular goals.

Thank You!!!