Priorities

Pankaj Rohal UNIT-2 1-1-2011

SATURDAY
JAN'11
1
WK-01 • 001-364

⇒ Representing knowledge using Rules :-

* Procedural Versus Declarative knowledge :-

1) → A declarative representation is one in which the knowledge is specified, but the use to which that knowledge is to be put is not given. To use a declarative representation, we must augment it with a program that specifies what is to be done to the knowledge and how.

"OR"

→ Declarative knowledge is descriptive representation of the knowledge expressed in factual statement.

→ Descriptive knowledge relates to a specific object. Includes information about meaning, roles, environment, resources, activities, associations and outcomes of the object.

for example, a set of logical assertions can be combined with a resolution theorem prover to give a complete program for solving problems.

"OR"

→ Declarative knowledge refers to the information that an expert knows from the different sources such as manuals or from the experience.

for example:- "All carnivorous have sharp teeth"
"Cheetah is a carnivore"

January 2011

2-1-2011

SUNDAY
JAN'11
2
WK-01 • 002-363

This can be represented using a declarative representation as:

$\forall x$ (carnivore (x) → sharp-teeth (x))
carnivore (cheetah)

using two statements, it is possible to deduce that "cheetah has sharp teeth".

2) **Procedural Representation**:- 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.

"OR"

A procedural representation, represents knowledge as procedure and inferencing mechanism manipulates these procedure to arrive at the result.

```
procedure carnivore (X);
    if (x = cheetah) then return true
        else return false
    end procedure carnivore (x).

procedure sharp-teeth (X);
    if carnivore (x) then return true
        else return false
    end procedure sharp-teeth (x).
```

To see whether cheetah has sharp teeth, one should activate the procedure with variable x instaniated to value cheetah. This procedure calls procedure Carnivore (x) in turn with value of (x = cheetah). Procedure Carnivore returns a value and so is procedure sharp-teeth.

3-1-2011

MONDAY
JAN'11
3
WK-02 • 003-362

for example, consider the knowledge base:

man (Aman)
man (Sunny)
person (Ram)
$\forall x$: man(x) → person(x)

Now consider trying to extract from this knowledge base the answer to the question

$\exists y$: person(y)

we want to bind y to a particular value for which person is true. Our knowledge base justify any of the following answer.

y = Aman
y = Sunny
y = Ram.

Because there is more than one value that satisfies the predicate, but only one value is needed, the answer to the question will depend on the order in which the assertions are examined. If we view the assertions as declarative, then they do not themselves say anything about how they will be examined. If we view them as procedural, then they do.

* **FORWARD and BACKWARD REASONING :-** In rule base system there is set of rules, which rule is to be followed is considered on the basis of two fundamental reasoning procedures or we can say two directions in which such a search could proceed :

1) Forward Reasoning or chaining or Data Driven Inferenc
2) Backward Reasoning or chaining or Goal — —.

Priorities

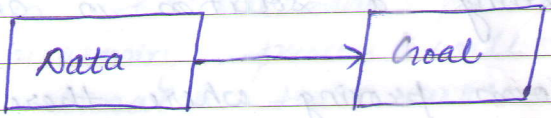
4-1-2011

TUESDAY
JAN'11
4
WK-02 • 004-361

1) **FORWARD REASONING OR CHAINING** :- In this the focus of attention starts with the known data. It is also known as data driven reasoning since input data are used to guide the direction of the reasoning.

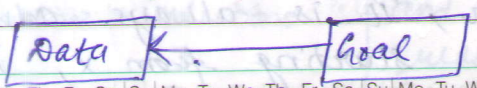
→ In it the problem solver begins with the given facts of the problem and a set of legal moves or rules for changing the states.

→ Search proceeds by applying rules to facts to produce new facts, which are in turn used by the rules to generate more new facts. This process continues until it generates a path that satisfies the goal condition.



2) **BACKWARD REASONING OR CHAINING** :- An alternative approach is possible i.e backward chaining. As the name suggests, it works backward from the goal. See what rules or legal moves could be used to generate this goal and determine what conditions must be true to use them. These conditions become the new goals (sub-goals) for the search.

→ Search continues, working backward through successive sub-goals until it works back to the facts of the problem. This finds the chain of moves or rules leading from a data to a goal, although it does so in backward order.



February	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa	Su	Mo	•	•	•
2011	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	•	•	•

5-1-2011

WEDNESDAY
JAN'11
5
WK-02 • 005-360

some search use both forward and backward reasoning depending upon type of problem and information available. The strategy is called bidirectional search.

⇒ **FACTORS DECIDING THE DIRECTION OF REASONING:-**

a) Are there more possible start states or goal states? one would like to move from smaller set of set of states to the larger (and thus easier to find) set of states.

→ for example, in chess game, there is one start state and many goal state, so it is advisable to use forward reasoning. There is bright chances of eventually getting a solution in any depth.

→ But in boolean theorm proving where there is one start state and one goal state, then search in any direction will be equally good.

b) In which direction is the branching factor (i.e avg. no. of nodes that can be reached directly from single node) greater? one would like to proceed in the direction with the lower branching factor and the reason is reduction in size of the search tree resulting in saving memory and processor time.

→ for example, if we will have to check the credentials of a person x claiming to be descendent of king Akbar, It is always advisable to generate a family tree starting from x, identifying his

6-1-2011

THURSDAY
JAN'11
6
WK 02 • 006-359

parents and then their parents and so on because the branching factor is too. But if we will start from King Akbar, identifying his children and their children then the size of tree generated will be large because branching factor will be larger than two.

c) Will the problem be asked ~~from~~ to justify its reasoning process to a user? If so, it is important to proceed in the direction that corresponds more closely with the way the user will think.

→ For example, an Expert system imitating the behaviour of an expert doctor, An expert system is supposed to explain its reasoning process to its user, and that reasoning process will be comprehensible if it will match the reasoning process of its user.

⇒ COMPARISON BETWEEN BACKWARD and FORWARD REASONING

	BACKWARD REASONING	FORWARD REASONING
→ Also called	Goal directed	Data Driven
→ Starts from	Possible solutions	New Data
→ Works towards	Necessary Data	Any conclusions
→ Style	conservative	opportunistic
→ processing	Efficient	Possibly wasteful
→ User impression	Plodding but predictable	Responsive but quirky
→ Obvious usage	Selection between alternative solutions	Building up solutions and "leaps" in reasoning.

7-1-2011

FRIDAY
JAN'11
7
WK-02 • 007-358

* MATCHING :- In rule base system there are set of rules. How we extract from the entire collection of rules those that can be applied at a given point. To do so requires some kind of matching between current state and preconditions of the rules

REASONS FOR USING MATCHING :-
→ It controls sequence of operations.
→ It identifies or classifies objects
→ Helpful in determining the best alternative.
→ To Retrieve items from a database.

AREA WHERE MATCHING IS USED :-
→ Speech Recognition
→ Natural Language Processing
→ Learning
→ Computer vision, experts systems
→ Automatic programming.

(a) ~~INDEXING~~ ~~Matching~~ :- One way to select applicable rules is to do a simple search through all the rules, comparing each one's preconditions to the current states and extracting all the ones that match. But there are two problems with this ~~match~~. simple solution :
(i) In order to solve very intersting problems, it will be necessary to use a large no. of rules. Scanning through all of them at every step of search would be hopelessly inefficient.

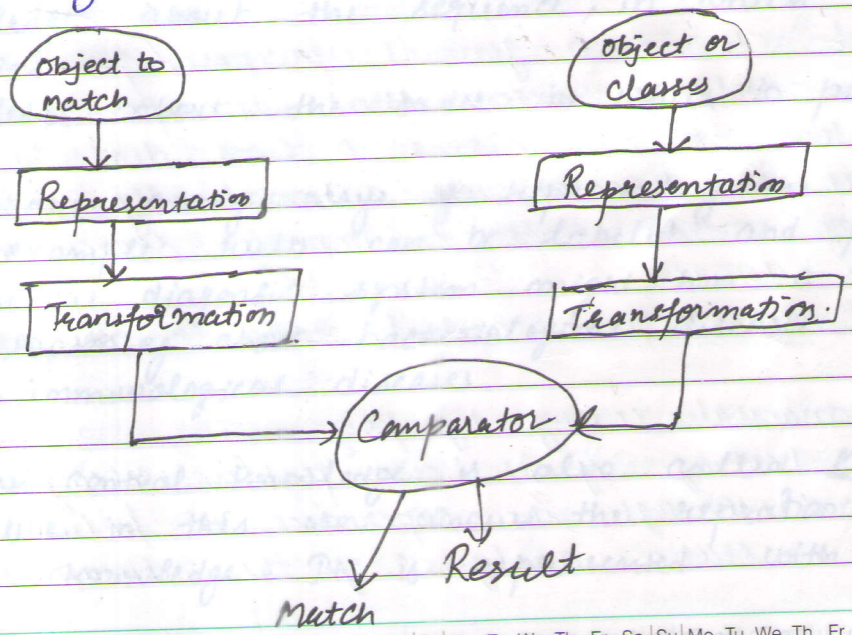
8-1-2011

SATURDAY
JAN'11
8
WK-02 •008-357

(ii) It is not always immediately obvious whether a rule's precondition are satisfied by a particular state.

Instead of searching through the rules, use the current state as an index into rules and select the matching one immediately.

for example, consider the legal move generation rule for the Chess. To be able to access the appropriate rules immediately, all we need do is assign an index to each board position. This can be done simply by treating the board description as a large number. Any reasonable hashing function can then be used to treat that number as an index into rules. All the rules that describe a given board position will be stored under the same key and will be found together.



MONDAY
JAN'11
10
WK-03 • 010-355

⇒ **CONTROL KNOWLEDGE** :- When there are many possible paths of reasoning it is critical that fruitless ones not to be pursued. Knowledge about which paths are most likely to lead quickly to a goal state is often called search control knowledge.

"OR"

The knowledge about a problem that is represented by the control strategy is often known as control knowledge. It includes knowledge about variety of process strategies, and structure used to coordinate the entity problem solving process. It can take many forms;

→ knowledge about which states are preferred instead to others.
→ knowledge about which rules to be applied in a given situation
→ knowledge about the sequence, in which rules are applied.
→ knowledge about the order in which to pursue subgoals.

There are many ways of representing the control knowledge for example rules can be labeled and partioned. A medical diagonsis system might have a set of rules for reasoning about bacteriological disease another set for immunological diseases.

* Search control knowledge is also called <u>meta knowledge</u> because in this we discuss the repration of knowledge about knowledge. It is represented with the help of rules.

TUESDAY
JAN'11
11
WK-03 • 011-354

The syntax for control rule is shown below:-

Under conditions A and B,
Rules that do {not} mention x
{ at all,
in their left-hand side
in their right-hand side }
will
{ definitely be useless,
probably be useless
....
probably be especially useful
definite be especially useful }.

There are number of AI system, which uses the control knowledge with rules. Examples of such systems are SOAR and PRODIGY.

a) PRODIGY:- It is a general purpose problem solving system that incorporates several different learning mechanisms. A Good deal of learning in PRODIGY is directed at automatically constructing a set of control rules to improve search in a particular domain.
PRODIGY can acquire control rules in no. of ways:-

→ Through manual coding by programmer.
→ Through looking of it own problem-solving behaviour.
→ It learn through its experience or failure.
→ Through a static analysis of the domain operators.

WEDNESDAY
JAN'11
12
WK-03 • 012-353

b) **SOAR** :- SOAR is general architecture for building intelligent systems. SOAR is based on a set of specific, cognitively motivated hypotheses about the structure of human problem solving. These hypotheses are derived from what we know about short-term memory.

In SOAR:

(i) Long-term memory is stored as a set of productions (rule)

(ii) Short-term memory (also called working memory) is a buffer that is affected by perceptions and serves as a storage area for facts deduced by rules in long-term memory.

(iii) There are different classes of problem solving activities.

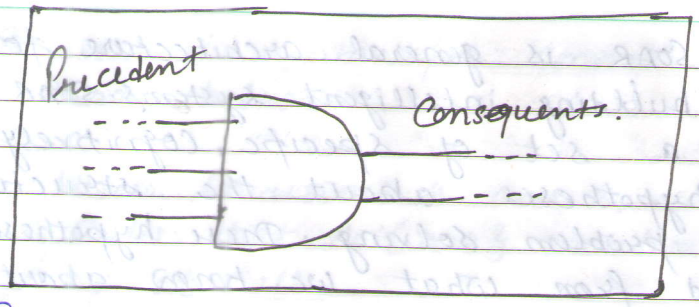
(iv) All intermediate and final results of problem are remembered for future refering.

Control knowledge can tell us something about the order in which we should pursue the goals. Suppose we are faced with the problem of building a piece of wooden furniture. The problem specify that wood must be sanded, sealed and painted. Which of there goal do we pursue first?

* **RULE-BASED DEDUCTION SYSTEM** :- Rule-Based problem solving systems are built using rules like the following, each of which contains several if pattern and one or more then patterns:

13-1-2011

THURSDAY
JAN'11
13
WK-03 • 013-352



In If if1
if2
!
then then1
then2.
!

A statement that something is true, such as "stretch has long legs" or "stretch is giraffe" is an assertion. In all rule based system each if pattern is a pattern that may match one or more of the assertions in a collection of assertions. The collection of assertion sometimes called working memory.

In many rule based systems, the then pattern specify new assertions to be placed into working memory, and the rule-based system is said to be a deduction system. In deduction system, the convention is referred to each if each pattern as an precedent and to each then pattern as a consequent.

Sometimes, however, the then patterns specify actions rather than assertions - for example:

14-1-2011

FRIDAY
JAN'11
14
WK-03 • 014-351

"Put the item into bag" - in this case the rule-based system is a reaction system.

In both deduction systems and reaction systems forward chaining is the process of moving from the if patterns to the then patterns using the if patterns to identify appropriate situations for the deduction of a new assertion or the performance of an action.

* APPROACHES TO KNOWLEDGE REPRESENTATION :-

→ **Representational Adequacy** :- The ability to represent all kinds of knowledge that are needed in that domain.

→ **Inferential Adequacy** :- It is the ability to manipulate the knowledge represented to produce new knowledge corresponding to that inferred from original.

→ **Inferential Efficiency** :- It is ability to direct the inferential mechanism into most productive directions by storing appropriate guides

→ **Acquisitional Efficiency** :- It is the ability to acquire new knowledge using automatic methods wherever possible rather than reliance on human intervention.

SATURDAY
JAN'11
15
WK-03 • 015-350

→ **Simple Relational Knowledge :-**

The simple way to represent declarative fact is as a set of relations of the same sort used in database systems.

Player	Height	Weight	Bats - Throws
A	5-10	170	R-R
B	6-0	180	R-R
C	6-2	190	L-L.
D	6-3	160	L-R
E	6-1	200	R-L

The reason that this representation is simple is that standing alone it provides very weak inferential capabilities but knowledge represented in this form may serve as the input to more powerful inference engines.

for example, in figure it is not possible even to answer the simple question, " Who is the haviest player? " But if a procedure for finding the haviest player is provided, then these facts coill enable the procedure to compute answer.

→ **INHERITABLE KNOWLEDGE :-**

The relational knowledge corresponds to a set of attributes and associated value that together describe the objects of the knowledge base. Knowledge about objects, their attributes, and their value need not be as simple as that shown in our

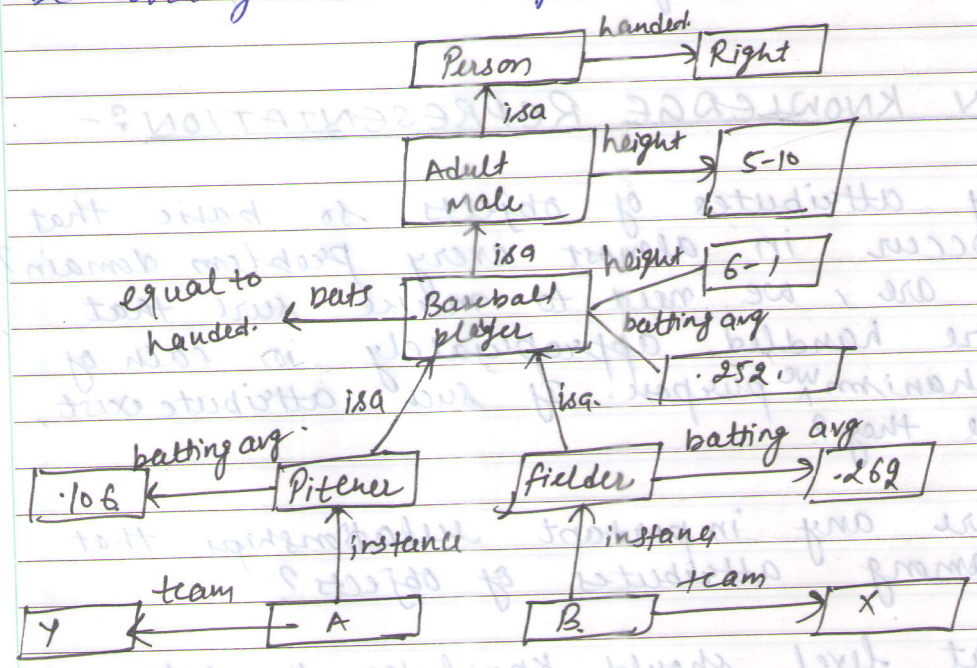
16-1-2011

SUNDAY
JAN'11
16
WK-03 • 016-349

example.

→ For this to be effective, the structure must be designed to correspond to the inference mechanisms that are desired. One of the most useful forms of inference is property inheritance, in which element of specific classes inherit attributes and values from more general classes in which they are included.

In order to support property inheritana, objects must be organized into classes and classes must be arranged in a generalization hierarchy.



~~In order to support property inheritance~~ figure shows some additional baseball knowledge inserted into a structure that is so arranged. Lines represents attributes. Boxed nodes represent objects and values of attributes of the objects.

MONDAY
JAN'11
17
WK-04 • 017-348

The structure shown in figure is a slot-and-filler structure. It may also be called semantic network or a collection of frames.

Baseball-Player
isa: Adult-male
bats: (Equal handed)
handed: 6-1
batting avg: .252.

figure:- Viewing a node as a frame.

* ISSUE IN KNOWLEDGE REPRESENTATION:-

- Are any attributes of objects so basic that they occur in almost every problem domain? If there are, we need to make sure that they are handled appropriately in each of the mechanism we purpose. If such attribute exist, what are they?
- Are there any important relationships that exist among attributes of objects?
- At what level should knowledge be represented
- How should sets of objects be represented?
- Given a large amount of knowledge stored in a database how can relevant parts be accessed when they are needed