جامـــــعـة المـــــستـقـبـل
AL MUSTAQBAL UNIVERSITY

كلية العلوم
قـســــــــم الانــظـــمـــة الـــطـبـيـة الــذكـــيــة

Lecture: (9)

Reasoning Systems

Subject: Artificial Intelligence

Class: Third

Lecturer: Dr. Maytham N. Meqdad

Reasoning Systems

1. Reasoning Systems

If-then rules have become the most popular form of declarative knowledge representation used in artificial intelligence applications. There are several reasons for this. Knowledge represented as if-then rules is easily understandable. Most people are comfortable reading rules, in contrast to knowledge represented in predicate logic. **Each rule can be viewed as a standalone piece of knowledge or unit of information in a knowledge base**. New knowledge can be easily added, and existing knowledge can be changed simply by creating or modifying individual rules.

There are several reasons for this:.

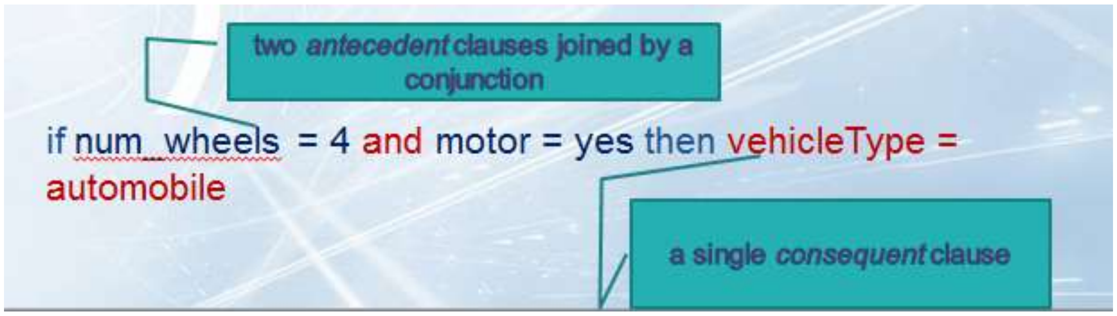
1- Knowledge represented as if-then rules is easily understandable.
2- Most people are comfortable reading rules, in contrast to knowledge represented in predicate logic.
3- Each rule can be viewed as a standalone piece of knowledge or unit of information in a knowledge base.
4- New knowledge can be easily added, and existing knowledge can be changed simply by creating or modifying individual rules.
5- Rules are easily manipulated by reasoning systems.
6- Forward chaining can be used to produce new facts (hence the term “production” rules).
7- backward chaining can deduce whether statements are true or not

Rule-based systems were one of the first large-scale commercial successes of artificial intelligence research. An ***expert system*** or ***knowledge-based system*** is the common term used to describe a rule-based processing system.
It consists of three major elements,

1- a *knowledge base* (the set of if-then rules and known facts),
2- a *working memory* or database of derived facts and data,
3- an *inference engine*,, which contains the reasoning logic used to process the rules and data.

1. Rules or Knowledge Base

– Unordered set of user-defined "if-then" rules.
– Form of rules: if P1, ..., Pm then A1, ..., An
– (Pm’s) are conditions (often facts) that determine when rule is applicable.
– (An) Actions can add or delete facts from the Working Memory.


two antecedent clauses joined by a conjunction
if num_wheels = 4 and motor = yes then vehicleType = automobile
a single consequent clause

2. Working Memory (WM)

– A set of "facts“, represented as literals, defining what are known to be true about the world
– WM initially contains case specific data (not those facts that are always true in the world)
– Inference may add/delete fact from WM
– WM will be cleared when a case is finished

3. Inference Engine

– Procedure for inferring changes (additions and deletions) to Working Memory.
– Can be both forward and backward chaining
– Usually a cycle of three phases: match, conflict resolution, and action)

A rule states a relationship between clauses (assertions or facts) and, depending on the situation, can be used to generate new information or prove the truth of an assertion.

Most rule-based systems allow rules to have names or labels such as

Rule1: or Automobile: to easily identify rules for editing or for tracing during inferencing. Some systems allow disjunctions (or) between antecedent clauses. This is a short-hand that reduces the size of a rule base. For example:

Rule 1: if num_wheels = 2 then vehicleType = cycle

Rule 2: if num_wheels = 3 then vehicleType = cycle

Rule 3: if (num_wheels = 2 or num_wheels = 3) then vehicleType = cycle

Rule 3: could replace Rule 1 and Rule 2 in the rule base.

Most rule systems also allow Boolean condition operators such as <, >, and !=, in addition to equality. We could rewrite Rule 3 without using disjunctions:

Rule 4: if (num_wheels > 1) and (num_wheels < 4) then vehicleType = cycle

Forward Channing

It is a data-driven reasoning process where a set of rules is used to derive new facts from an initial set of data. - It does not use the resolution algorithm used in predicate logic.

- The forward-chaining algorithm generates new data.

- As an inference procedure, forward chaining is very fast.
- Forward chaining is also used in real-time monitoring and diagnostic systems where quick identification and response to problems are required.
- any expert system requires three basic elements,
- 1- a knowledge base of rules and facts,. 2- a working memory for storing data during inferencing 3- an inference engine.

Forward Chaining Algorithm

1. Load the rule base into the inference engine, and any facts from the knowledge base into the working memory.

2. Add any additional initial data into the working memory.

3. ***Match*** the rules against the data in working memory and determine which rules are triggered, meaning that all of their antecedent clauses are true . This set of triggered rules is called the ***conflict set.***

4. Use the ***conflict resolution procedure*** to select a single rule from the conflict set.

5. Fire the selected rule by evaluating the consequent clause(s); either update the working memory if it is a fact-generating rule, or call the ***effector procedure***, if it is an ***action rule***. This is referred to as the act step.

6. Repeat steps 3, 4, and 5 until the conflict set is empty.

Forward Chaining (Conflict resolution Procedure)

- **Select the first rule in the conflict set**. This is certainly simple, and for some domains it works.
- **Select the rule with the highest specificity or number of antecedent clauses**. The idea here is to select the rule that is the most specific (has the most test conditions on the if part of the rule) before we fire more general rules that have fewer antecedent clauses.
- **Select the rule that refers to the data which has changed most recently**. This method requires that changes to the working memory are **time-stamped** or somehow **tagged** to show when they were last modified.

If the rule has fired in the previous cycles, don't add it to the conflict set, this rule some time extended to limit rule , so they can fire only once.

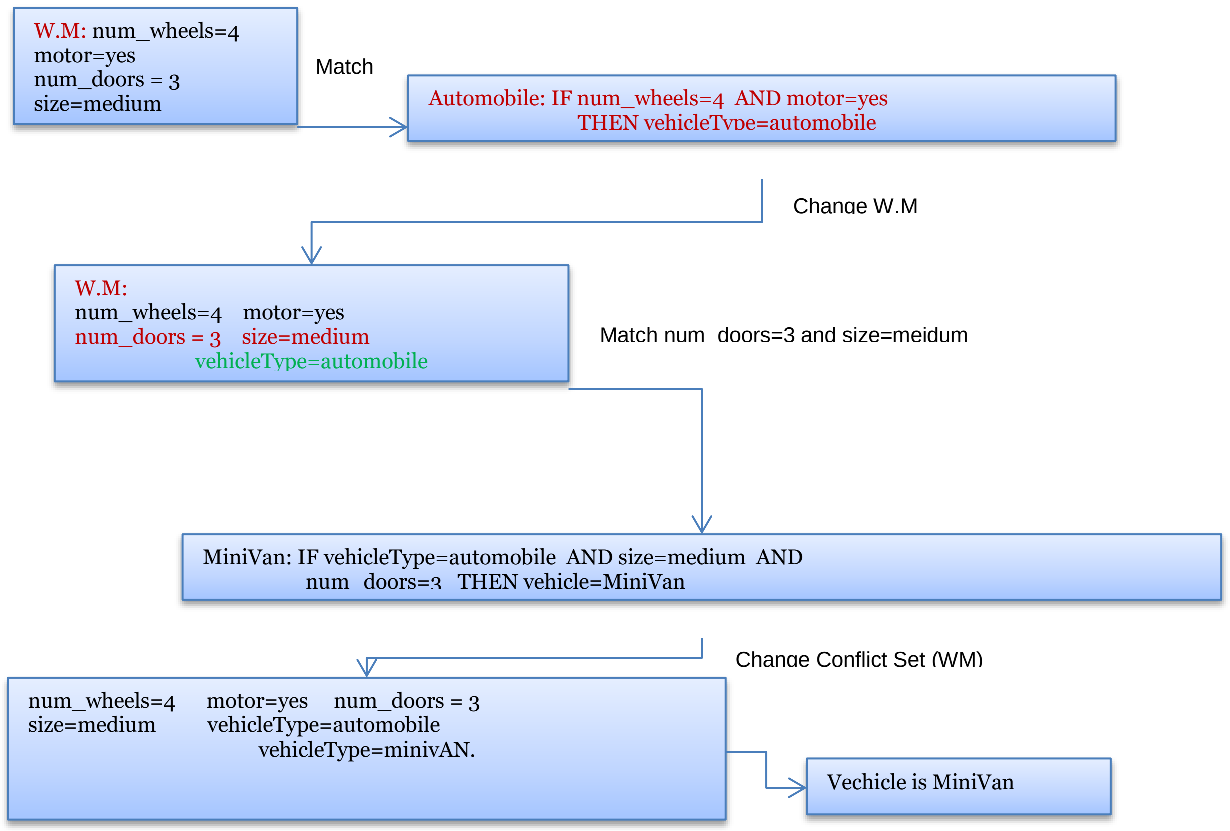
A common control strategy in rule-based systems is to assign priorities to rules which can be used to aid in the selection process. If we have a process that proceeds in three phases, we can assign priority 1 to the set of rules that contribute to the first phase, priority 2 to the rules in the second phase, and likewise for the third subset of rules. The advantage of using priorities is that it greatly reduces the number of rules that have to be searched and tested in the match phase.

Another approach that can be used to achieve the same results, even if the inferencing system doesn't formally support priorities, is to use guard clauses. For example, we could add a clause priority = 1 to the antecedents of all rules in group one, priority = 2 in group two and so on. While less efficient than if the inferencing system supports priorities, it does produce the desired behavior.

Vehicles rule base:-

```
Bicycle: IF vehicleType=cycle
               AND num_wheels=2
               AND motor= no
              THEN vehicle= Bicycle
Tricycle: IF vechicleType=cycle
                AND num_wheels=3
                AND motor=no
               THEN vehicle=Tricycle
Motorcycle: IF vehicleType=cycle
                     AND numb-wheels=2
                     AND motor=yes
                THEN vechicle=Motorcycle
```

SportsCar: IF vehicleType=automobile
AND size=small
AND num_doors=2
THEN vechicle=Sports_Car
Sedan: IF vechicleType=automobile
AND size=medium
AND num_doors=4
THEN vehicle =Sedan
MiniVan: IF vechicleType=automobile
AND size=medium
AND num_doors=3
THEN vechicle=MiniVan
SUV: IF vechicleType=automobile
AND size=large
AND num_doors=4
THEN vehicle=Sports_utility_vechicle
Cycle: IF num_wheels < 4
THEN vechicleType= cycle
Automobile: IF num_wheel=4
AND motor=yes
THEN vechicleType=automobile


W.M: num_wheels=4
motor=yes
num_doors = 3
size=medium
Match
Automobile: IF num_wheels=4 AND motor=yes
THEN vehicleType=automobile
Change W.M
W.M:
num_wheels=4 motor=yes
num_doors = 3 size=medium
vehicleType=automobile
Match num doors=3 and size=meidum
MiniVan: IF vehicleType=automobile AND size=medium AND
num doors=3 THEN vehicle=MiniVan
Change Conflict Set (WM)
num_wheels=4 motor=yes num_doors = 3
size=medium vehicleType=automobile
vehicleType=minivAN.
Vechicle is MiniVan