Rules

Carlos García Martínez

Outline

- Introduction
- Basic Components of Rule Based Systems (RBS)
- Rule structure
- Inference mechanism
- Reasoning control
- Reasoning explanation
- Uncertainty
- Discussion. Conclusions

Preliminaries

- Rules in Computer Science:
  - Production rules for **formal languages**.
  - Rules that create **state spaces** in search problems.
- First RBSs appear in 70s:
  - [Newell and Simon, 1972] They model intelligent behaviour by means of rules (an intelligent agent behaves according to some rules).
  - [Buchanan and Feigen, 1978] First RBS. It produces chemical structures explaining expetographic results.
  - [Buchanan and Shortliffe, 1984] MYCIN. First RBS that applies rules in the way we understand them, at present.

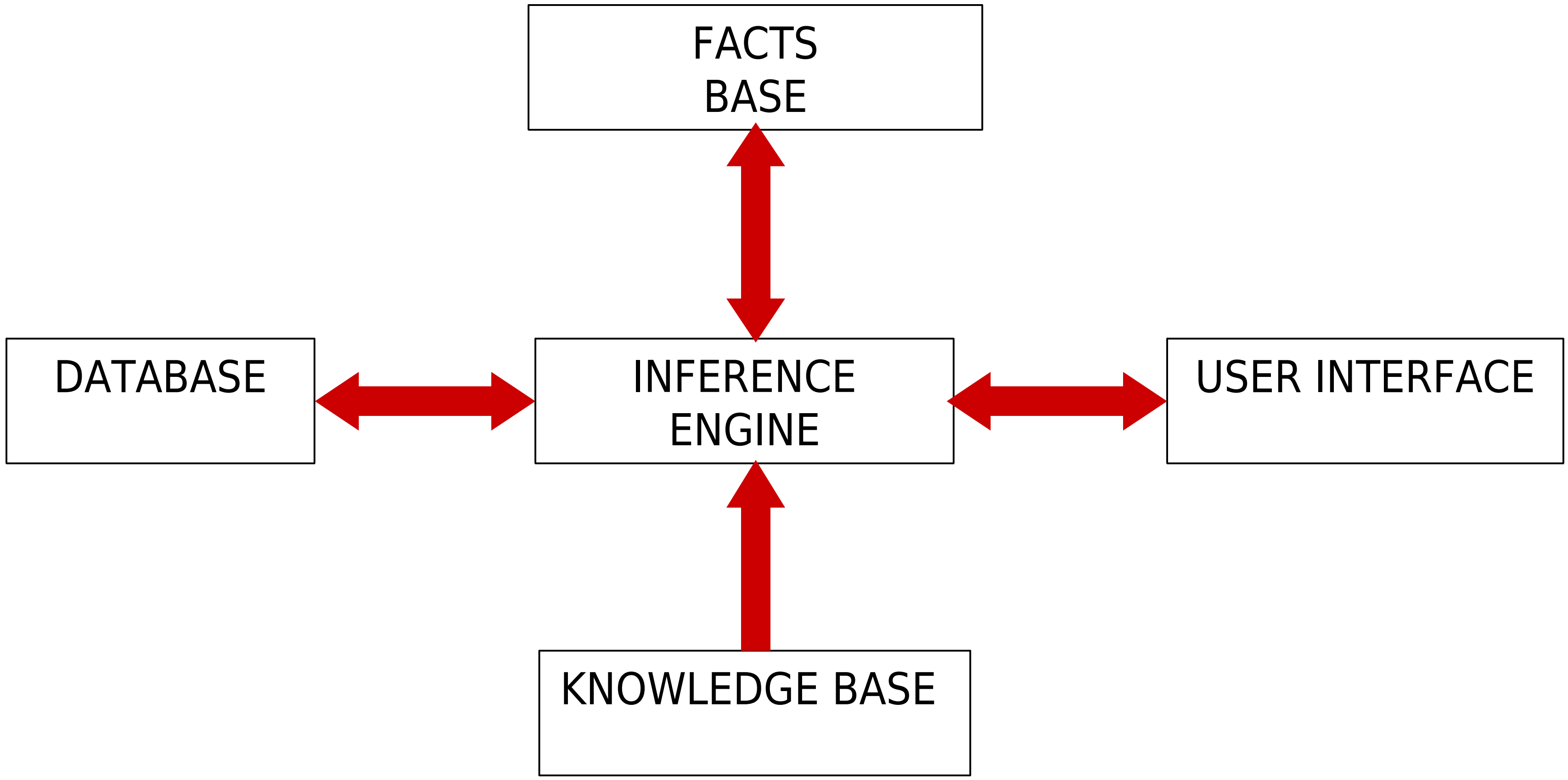
Concepts

- ***RBS*** analyse data and ask for new information till reaching a diagnostic
- Though they generate a search space, it is not relevant
- Nowadays, ***Expert Systems*** make use of rules, frames, logic and concepts from other ontology languages

Outline

- Introduction
- ***Basic Components of Rule Based Systems (RBS)***
- Rule structure
- Inference mechanism
- Reasoning control
- Reasoning explanation
- Uncertainty
- Discussion. Conclusions

Basic Components



Outline

- Introduction
- Basic Components of Rule Based Systems (RBS)
- ***Rule structure***
- Inference mechanism
- Reasoning control
- Reasoning explanation
- Uncertainty
- Discussion. Conclusions

IF and THEN Parts

- General structure of a rule:

  IF ... THEN ...

- **IF part** lists a set of clauses, in some logic combination, that has to be fulfilled.
- **THEN part** shows conclusions that can be inferred (***declarative programming***) or actions that the system has to perform to apply the rule (***imperative programming***).

Kind of Clauses (IF part)

- **Hypothesis**:
  - Fire, hypertension, diabetes, there are firemen, road is frozen, etc.
- **Comparison relations**:
  - body temperature > 39
  - Juan.age = 4
- **Membership functions**:
  - Panama **IS** country
  - Juan **IS** person

Actions in THEN part

- **To assert** new facts
- **To retract** previous facts
- **To perform an action** (to print in the screen)

IF

(inner *temperature > wanted temperature)*

AND

(wanted temperature > outer temperature)

THEN

PERFORM turn on fan

ASSERT fan = on

Variables

- Improve rule **expressiveness,** though is higher in Predicate Logic
- Example: All humans are mortal

IF *x IS human*

THEN *ASSERT x IS mortal*

$$\forall x\ (human(x) \rightarrow mortal(x))$$

Other characteristics

- Higher expressiveness lead to complex inference engines
- How to state that fact A is false:
  - You can **explicitly** tell the computer if fact A is true, false, not evaluated, unknown (Nexpert).
  - ***Closed world axiom***: any statement that is not known to be true is false (Prolog).
  - Designer can express if it is false or unknown (GoldworKs).
- ***Univalue and Multivalue Variables***
  - ***Univalue***, new assignment replace old values (*Marta.age = 4*).
  - ***Multivalue***, new assignment do not replace old values (Marta.languages += 'English')

Outline

- Introduction
- Basic Components of Rule Based Systems (RBS)
- Rule structure
- ***Inference mechanism***
- Reasoning control
- Reasoning explanation
- Uncertainty
- Discussion. Conclusions

Pattern Matching

- IF parts often use ***patterns:***
  - **Clauses without variables:** they are “active” if the required facts are in the fact base. They provide one rule instance.

IF *rain AND distance > 1 km*

THEN *ASSERT advise =* '*take umbrella*'

  - Clauses with variables: they can be active many times, as much as sets of facts can fulfill IF part.

IF *x IS teacher AND*

x.knowledge area = Computer Science

THEN *ASSERT x.can teach = Applied Computing*

Chaining

- Executing rules activates other rules.
- Two types:
  - **Forward chaining**: facts driven
  - **Backward chaining**: goal driven. It looks for the rule that express if the goal is true or false (Y/O graphs). It often ask the user the information that can not be deduced
- Backward chaining performs a more specific search than forward chaining
- *Rules always execute from IF part toward THEN part. It does not depend on chaining*

Logical dependence

- Differing from Logic, RBS can retract information from the fact base (something that was true is now false)
- Retracting facts may affect other facts.
- Dependence:
  - **Reversible (Logical dependence)**: retracting facts affects other facts

    IF *light = on* ***THEN*** *room is lit*

  - **Irreversible**: retracting facts does not affect other facts

    IF *light = on* ***THEN*** *film get fogged*

Outline

- Introduction
- Basic Components of Rule Based Systems (RBS)
- Rule structure
- Inference mechanism
- ***Reasoning control***
- Reasoning explanation
- Uncertainty
- Discussion. Conclusions

Reasoning control

- It selects the **rule to be executed** among the all the active ones

- **Significance**:
  - It affects the results
  - It affects the efficiency
  - It affects communication between the system and user

Mechanisms

- To improve ***efficiency***:
  - Clause distribution within the IF part
  - Clauses controlling inference
- To control reasoning:
  - ***“Simplest”*** method: **Rule sorting**
  - An agenda
  - Several agendas (sponsors / modules)
  - Rule Sets
  - Meta-rules
  - Sensitivity and stability concepts

Clause Distribution

- Put first more restrictive clauses
- It enhances the procedure in charge of looking for active rules
- It does not affect the results

Clauses controlling Inference

- To add clauses pointing out the rules that should be activated and which not

    IF *state = car started*
    road frozen
    speed > 70
    THEN
    ASSERT advise = reduce speed

- It enhances efficiency, may control reasoning, and make debugging easier
- The problem is divided in several ***stages.*** In medicine:
    - state = 'to obtain history'
    - state = 'to diagnose'
    - state = 'to treat illness'

Rule Sorting

- Rules are sorted according to ***priority***. The first activated rule is executed.

- ***Drawbacks***:
  - It shows low elegance
  - Sorting might be time consuming
  - It is only applicable in simple systems

Agenda

- There are two phases:
  - Active rules are annotated in the agenda
  - Only one active rule is selected for execution
- The agenda store ***instances*** of active rules
- Agenda may be implemented as a list, stack, or queue, with or without priorities
- ***Priorities*** can be implemented by means of a numeric value, which can be modified in runtime.

Several Agendas

- ***There are several agendas,*** each one is controlled by one ***“sponsor” / “module”***
- Rules are related to one sponsor, whose agenda stores its instances
- Resources are distributed among sponsors
- The execution of a rule may make active or inactive rules within other agendas
- There is only one agenda with the focus at a time

Rule Sets

- They can be turned active or deactivate
- It is similar to introducing clauses controlling inference
- They do not possess different agendas
- Sponsors and rule sets let the designer to ***organize the knowledge base***, making the system construction and maintenance easier, and improving efficiency

Meta-rules

- They depict knowledge about the domain knowledge (***meta-knowledge***)
- They may organise shared attributes among different rules
- They may ***guide knowledge acquisition*** by means:
  - Model understanding
  - Learning by experience
- They store information about
  - Utility of the rules
  - Rule priorities

Example

IF *state = 'assessing qualification'*
r1 IS RULE
r1.thenPart contains 'assess qualification'

THEN

ASSERT r1.priority = -100

Domain dependent

IF *state = x*
r1 IS RULE
r1.thenPart contains x
THEN

ASSERT r1.priority = -500

Domain independent

Advantages of Meta-Rules

- Knowledge is more explicit
- They can modify the control strategy without modifying the inference engine
- They can show the reasons for executing one rules before others
- They reorganize knowledge
- They can learn from experience or from communication with user
- They can show what they know
- They help the inference engine guiding the chaining

Sensitivity and stability

- An intelligent system should be:
  - **Sensitive**: it should respond to new information
  - **Stable**: reply should be properly delimited (small changes in the input result in small changes in the output)
- They are opposing characteristics.

Sensitivity and Stability Mechanisms

- **Refraction:** An executed rule can not be active by means of the same facts. It favours stability

- **Actuality:** Rules active by means of recent facts are preferred. It favours sensitivity

- **Specificity:** Specific rules are preferred upon general ones. It often counts the number of clauses in the rules

Outline

- Introduction
- Basic Components of Rule Based Systems (RBS)
- Rule structure
- Inference mechanism
- Reasoning control
- ***Reasoning explanation***
- Uncertainty
- Discussion. Conclusions

Reasoning explanation

- ***Meta-rules*** can give information about the performed reasoning
- ***MYCIN*** let user to ask:
  - ***why*** the system require some information
  - ***how*** the system have reached a conclusion (executed rules)
  - This information was very useful for the designer. Users did not use functionality.
- ***NEOMYCIN*** [Clancey, 1984] contains meta-rules that can perform explanations about the strategic knowledge

Outline

- Introduction
- Basic Components of Rule Based Systems (RBS)
- Rule structure
- Inference mechanism
- Reasoning control
- Reasoning explanation
- ***Uncertainty***
- Discussion. Conclusions

Uncertainty

- A RBS should manage uncertainty because:
  - Information use to be ***imprecise/vague***, ***incomplete***, and sometimes, ***wrong***
  - Models can be ***incomplete***
  - The user domain is rarely deterministic
- How to manage uncertainty:
  - Fuzzy Logic
  - ***Certainty Factors***

Certainty Factors

- ***FC(e,h)*** outline how ***reliable hypothesis*** *h* is given evidence *e*
- They are similar to, but different, the probability of applying each rule
- They can be defined by means of:
  - Probabilities (a priori or conditioned)
  - Subjective values

Outline

- Introduction
- Basic Components of Rule Based Systems (RBS)
- Rule structure
- Inference mechanism
- Reasoning control
- Reasoning explanation
- Uncertainty
- ***Discussion. Conclusions***

RBS vs. Imperative Programming

- ***Style***:
  - Imperative programming is based on commands
  - RBS state knowledge that should be applied in each case (declarative programming)
- ***Execution:***
  - Commands in imperative programming are rigid and sequentially executed
  - Rule order is not significant
  - Reasoning is controlled by the inference engine
  - Chaining has no meaning in imperative programming
- ***Pattern matching:***
  - Rules are instantiated according to the facts that match their patterns (There may be several instances per rule)
  - Commands are not instantiated

RBS vs. Imperative Programming

- ***Actions can be undone:***
  - RBS can apply reversible and irreversible dependence
  - Commands in imperative programming can not be undone
- ***Explicit knowledge:***
  - RBS can explain reasoning, reorganise knowledge, and even learn from mistakes
- ***Uncertainty:***
  - RBS can manage uncertainty
  - Imperative Programming can not

RBS vs. Logic

- Both share several characteristics (***declarative programming***)
- Rules are not just data, but the program
- ***Rules reduce expressiveness and inference adequacy***, but are more efficient
- In general, rules can not use ***universal quantifiers***
- ***Modus ponens,*** from Logic, is not applicable in RBSs
- Chaining (rules) is less powerful than resolution principle (Logic), but more efficient

Criticisms

- They are not purely declarative. Results depends on the reasoning control applied
- Expressiveness is in contrast to efficiency
- ***The results is more unpredictable*** than using imperative programs.
- That leads to ***complex development and maintenance*** (even more with large systems)
- Uncertainty management is debatable