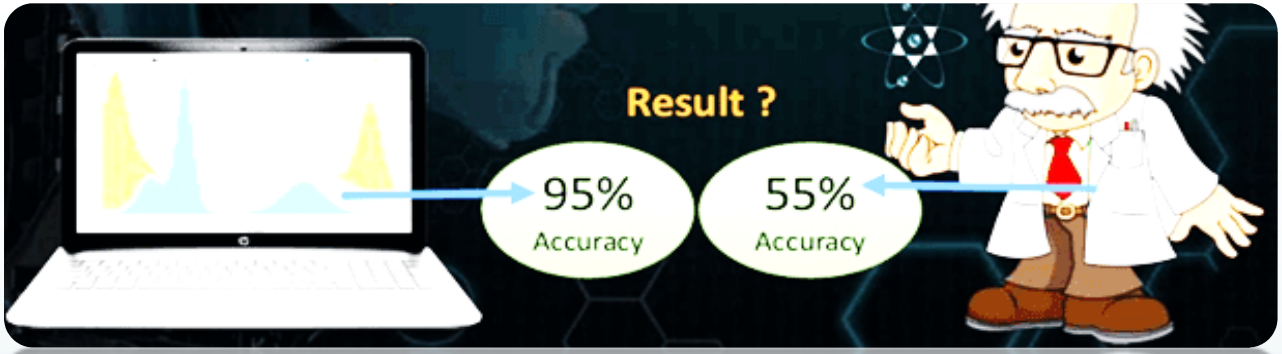


Introduction to Expert System

Dr Pradipta Biswas, *PhD (Cantab)*
Assistant Professor
Indian Institute of Science
http://cambum.net/

Contents

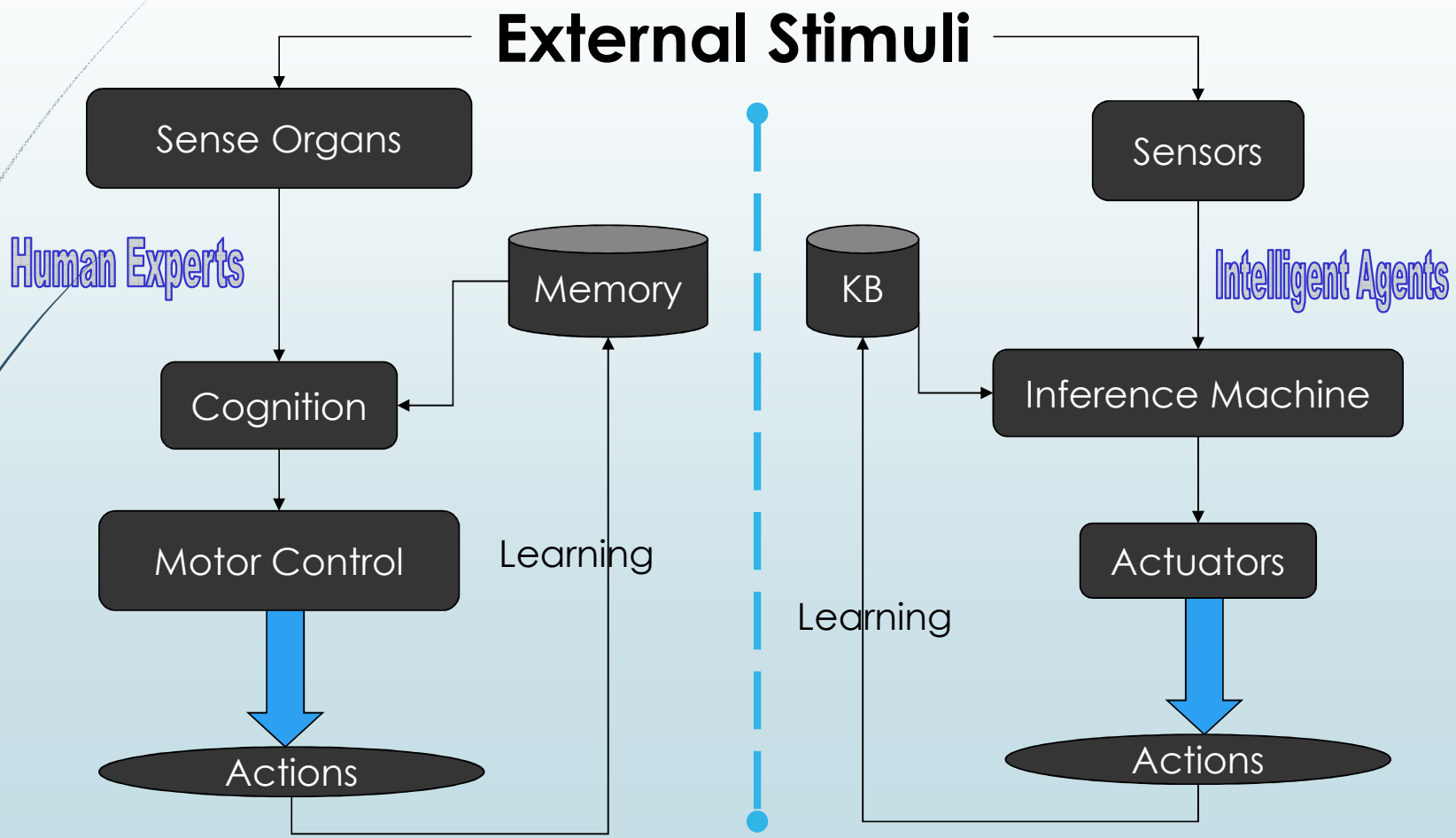
Expert & Knowledge-Based Systems

- Large-scale problem solving systems
  - Originally called expert systems, they would mimic the problem solving processes of domain experts
- Expert systems were originally developed by hand and most commonly in Lisp
- It was discovered that many problems were being solved by chaining through rules (if-then statements) that would operate on a collection of facts and partial conclusions
- These rule-based systems led to the first AI tools or shells
  - Today, to simplify expert system creation, most people use these AI shells – you just fill in the knowledge, the problem solving processes are already implemented

What is Reasoning

- Reasoning is the act of using reason to derive a conclusion from certain premises. There are two main methods to reach a conclusion.
- Intelligent: endowed with the capacity to reason

The Process of Reasoning



Reasoning Techniques

- Analogical Reasoning
- Deductive Reasoning
- Inductive Reasoning
- Formal Reasoning
- Procedural Reasoning….

Reasoning Techniques

- Formal Logic
- **Rule based System**
- **Case Based Reasoning**
- Frame based System

Trade-off :

Expressiveness vs. Efficiency of Reasoning

Historical Systems

Dendral

- The Dendral system (DENDRitic ALgorithm) was the first expert system, developed in the 1960s
  - Determines chemical composition from mass spectrogram data
- The approach: plan-generate-and-test with human feedback
- This is a constrained search technique
  - Generate a hypothesis: a possible chemical compound
  - Test the hypothesis: use a series of heuristics and subprograms to determine if the chemical compound generated is plausible given the data
  - If so, show it to the user – the user can *steer* the direction that DENDRAL takes next by suggesting what chemical elements should be added or removed from the generated hypothesis compound
  - DENDRAL repeats until an answer is found acceptable that does not violate the constraints as dictated by the data

Eliza

- Created by Joseph Weizenbaum at MIT
- Weizenbaum wrote the program so that it would generate an English response/question based on a group of patterns
- The most famous script, DOCTOR, simulated a Rogerian psychotherapist (in particular, Carl Rogers, who was well-known for simply parroting back at patients what they had just said), and used rules, dictated in the script, to respond with non-directional questions to user inputs.
- As such, ELIZA was one of the first chatterbots and one of the first programs capable of attempting the Turing test.

- Sample dialog:
  - H: Men are all alike
  - E: In what way?
  - H: They're always bugging us about something or another.
  - E: Can you be more specific?
  - H: Well, my boyfriend made me come here.
  - E: Your boyfriend made you come here?
  - H: He says I'm depressed much of the time.
  - E: I am sorry to hear you are depressed.
  - H: It's true. I am unhappy.
  - E: Do you think coming here will help you not to be unhappy?
  - H: I need some help, that much seems certain.
  - E: What would it mean to you if you got some help?
  - H: Perhaps I could learn to get along with my mother.
  - E: Tell me more about your family

MYCIN

- Implemented in the early 1970s, Mycin is perhaps the most recognized and cited expert system
- MYCIN was developed over five or six years in the early 1970s at Stanford University. It was written in Lisp as the doctoral dissertation of Edward Shortliffe under the direction of Bruce G. Buchanan, Stanley N. Cohen and others.
- MYCIN operated using a fairly simple inference engine and a knowledge base of ~600 rules.
- MYCIN received an acceptability rating of 65% on treatment plan from a panel of eight independent specialists
- MYCIN was never actually used in practice

Rule Based System

Rule Based Systems

- Rule looks like
  - If
    - Cond1
    - Cond2
    - …
  - Then
    - Then1
    - Then2
    - …

```
(rule (if has-sauce is yes and
          sauce is sweet)
      (then best-sweetness is sweet with certainty 90 and
            best-sweetness is medium with certainty 40))

(rule (if preferred-sweetness is dry)
      (then best-sweetness is dry with certainty 40))

(rule (if preferred-sweetness is medium)
      (then best-sweetness is medium with certainty 40))

(rule (if preferred-sweetness is sweet)
      (then best-sweetness is sweet with certainty 40))

(rule (if best-sweetness is sweet and
          preferred-sweetness is dry)
      (then best-sweetness is medium))

(rule (if best-sweetness is dry and
          preferred-sweetness is sweet)
      (then best-sweetness is medium))
```

Vocabulary/Terminology

- Assertion: A statement about a fact
- If-patterns match assertions in a collection of assertions called a
- Working Memory
- Deduction system: then-patterns specify assertions to be placed in working memory
- Reaction systems: then-patterns specify actions

Vocabulary/Terminology

- In deduction systems
  - Antecedent: if-pattern
  - Consequent: then-pattern
- Forward chaining: process of moving from if-patterns to then-patterns, using the if-pattern to identify appropriate situations for the deduction of a new assertion or the performance of an action

Deduction Systems

- Satisfying an assertion: When an if-pattern matches an assertion
- Rule Triggering: When all if-patterns of a rule are satisfied
- Rule Firing: When a triggered rule establishes a new assertion or performs an action

Zookeeper

- Identifies animals in a small zoo
- Robbie the robot can perceive
  - Color, size, hair, gives milk, …
- Can tell that an object is an animal, but cannot tell what animal it is
- Would be nice to write Robbie an animal identification system

Zookeeper Designs

- One if-then rule for each animal in the zoo
- If-then rules produce intermediate assertions
  - Only a few antecedents each. Easier
  - Forward chaining through intermediate assertions to identity of animal

Rules

- Small zoo
  - Tiger, cheetah, giraffe, zebra, ostrich, penguin, albatross
  - Zookeeper is simpler
- Z1:
  - If
    - ?x has hair
  - Then
    - ?x is a mammal

Variables and Bindings

- Antecedents and consequents contain variables (?x)
- Variables acquire values during the matching process

- Assertion in WM: Animal1 has hair
- ?x has hair matches when ?x becomes Animal1
  - Animal1 has hair
  - ?x has hair
- ?x is bound to Animal1 or
- Animal1's is ?x's binding

Bindings

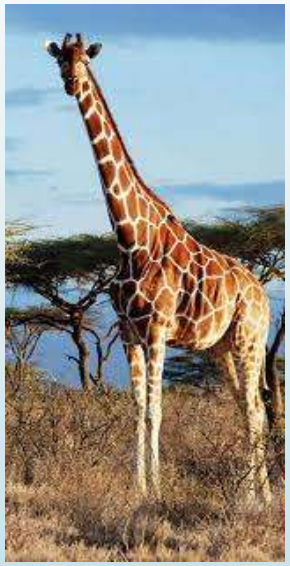
- Once a variable is bound, that variable is replaced by its binding wherever it appears in the same or subsequently processed patterns
- Whenever the variables in a pattern are replaced by their bindings, the pattern is said to be instantiated

Instantiation

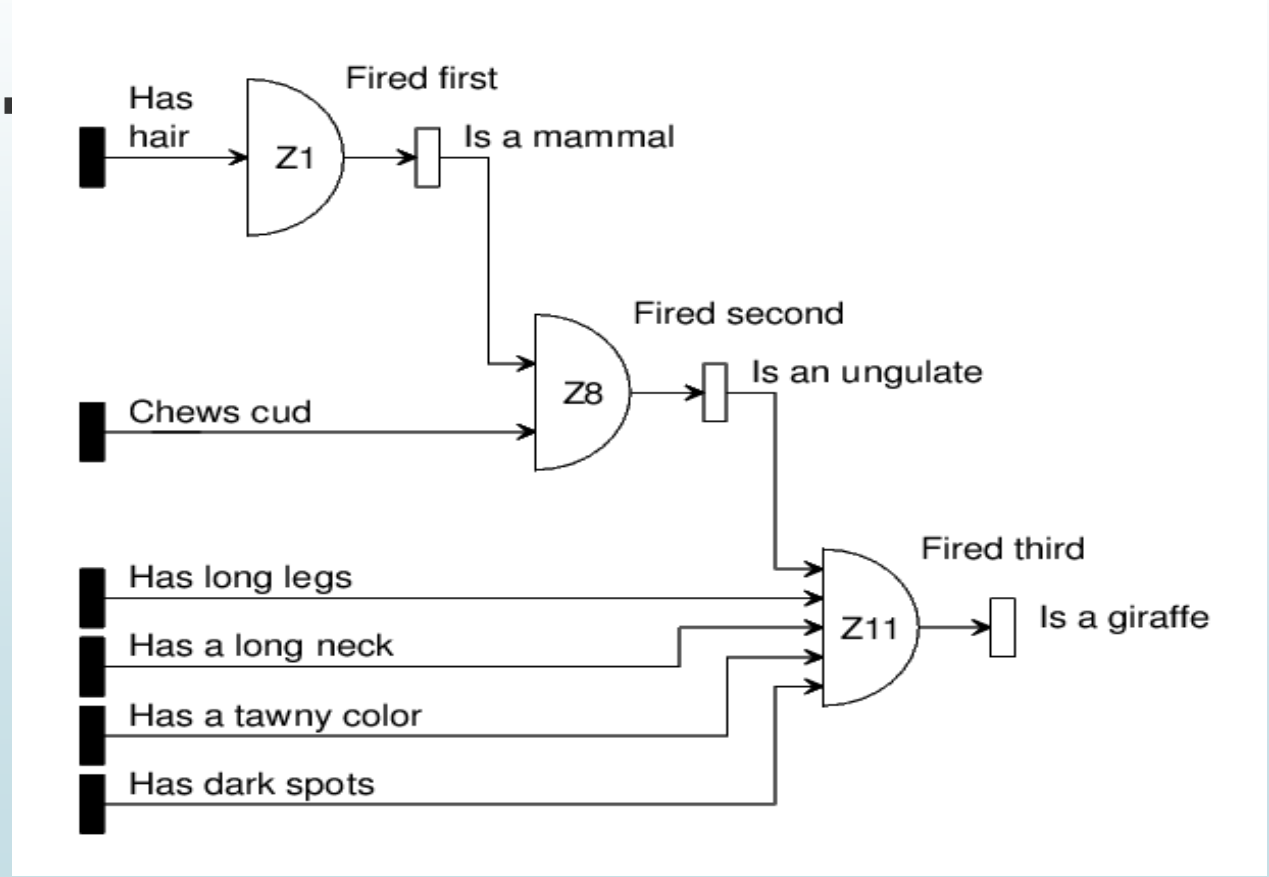
- Animal1 has hair
- If ?x has hair
- then ?x is-a mammal
- ?x bound to Animal1, then instantiated rule is
- If Animal1 has hair
- Then Animal1 is-a mammal

Working Memory

- Animal1 has hair
- Animal1 chews cud
- Animal1 has long legs
- Animal1 has tawny color
- Animal1 has dark spots
- Animal1 has a long neck



Rule Firing Sequence



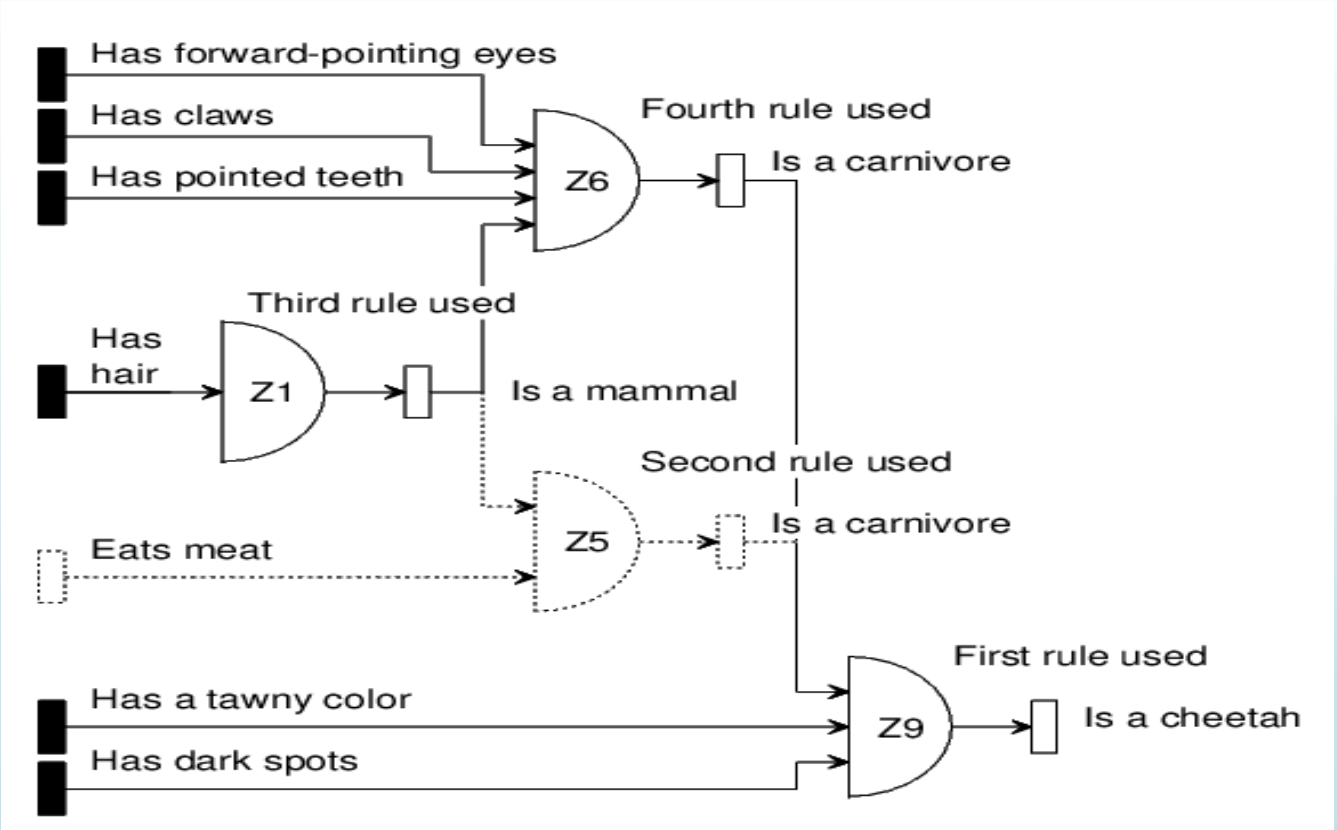
Forward Chaining

- Until no rule produces an assertion or the animal is identified
  - For each rule
    - Try to support each of the rule's antecedents by matching it to known assertions
    - If all the rule's antecedents are supported, assert the consequent unless there is an identical assertion already
    - Repeat for all matching and instantiation alternatives

Backward Chaining

- Form a hypothesis – rule consequent
- Work to find supporting assertions in rule antecedents
  - Animal2 has tawny color
  - Animal2 has dark spots
  - Animal2 has hair
  - Animal2 has forward pointing eyes
  - Animal2 has claws
  - Animal2 has teeth

Rule Chaining Backward



Backward chaining code

- Until all hypothesis have been tried and none have been supported or until the animal has been identified
  - For each hypothesis
    - For each rule whose consequent matches the current hypothesis
      - Try to support each of the rule's antecedents by matching it to assertions in WM or by backward chaining through another rule, creating new hypotheses. Be sure to check all matching and instantiating alternatives
      - If all the rule's antecedents are supported, announce success and conclude that the hypothesis is true

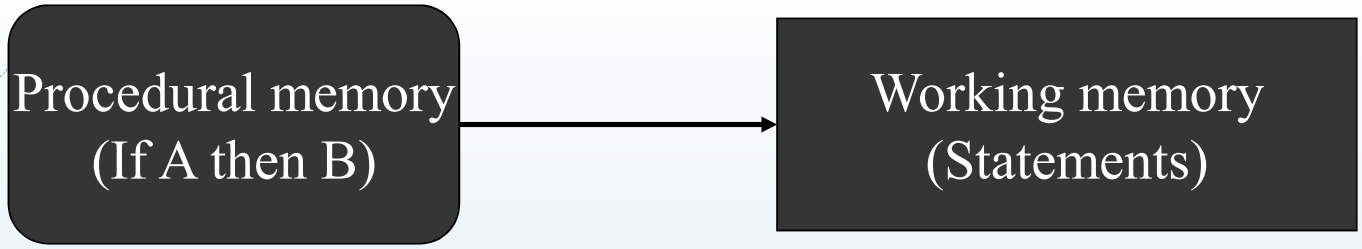
Forward or backward

- Depends on the problem!
- Fan-out versus Fan-in
- If the facts can lead to a large number of conclusions, but the number of ways to reach the particular conclusion we are interested in is small, then there is more fan-out than fan-in. **Use backward chaining**
- If the number of ways of reaching the particular conclusion in which you are interested is large, but the number of conclusions you are likely to reach using the assertions is small, you have more fan-in. **Use forward chaining**

Forward or Backward

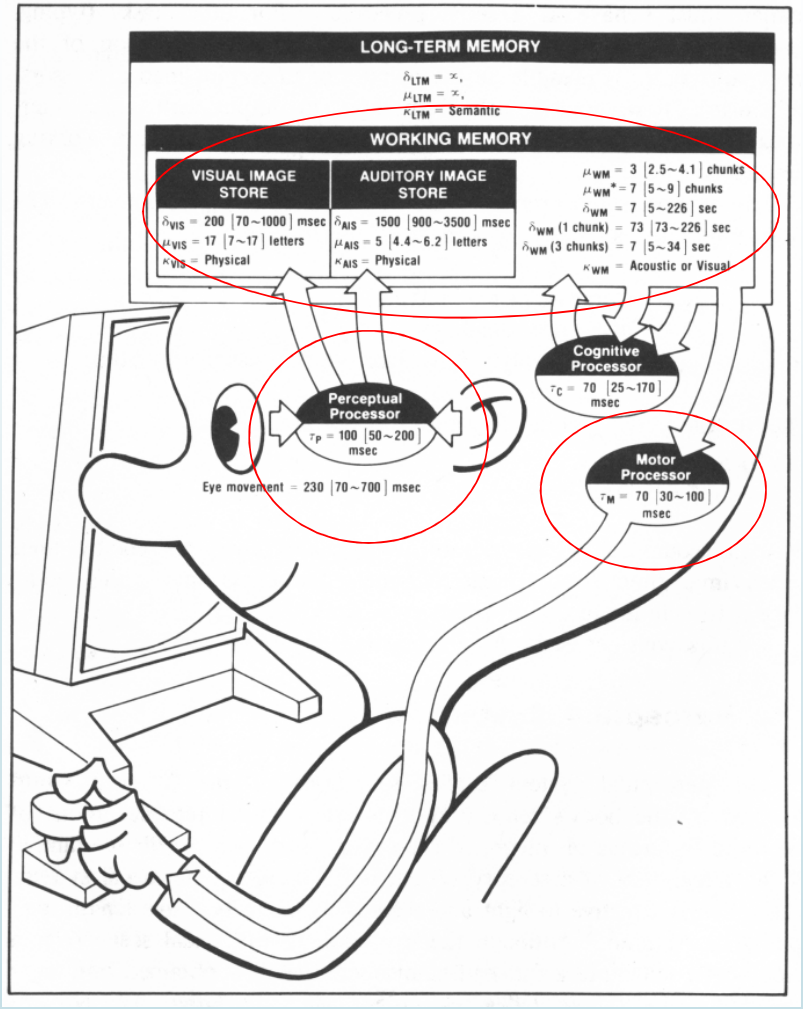
- If fan-in an fan-out are about the same
- If you have not yet gathered any facts, and if you are only interested in whether one of many possible conclusions is true, use backward chaining
- If you are only interested in whether an animal is a carnivore, only look at antecedents of carnivore rule to focus fact finding
- If you have all the facts that you will ever get, and are interested in everything that you can conclude from those facts, use Forward chaining
- If you catch a fleeting glimpse of an animal, gathered a set of facts. No more facts possible because animal is gone: Forward chain

SOAR - Symbolic system

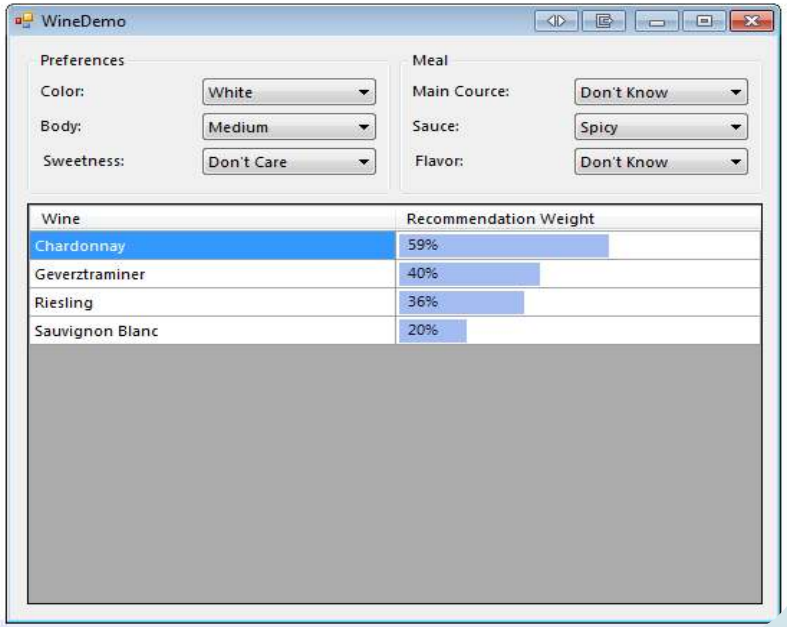


- State space model → Memory and problem definition
- Impasse → Can't find a rule, decompose goal to sub-goals
- Chunking → Learning new situation

Model Human Processor



```
(rule (if has-sauce is yes and
          sauce is spicy)
      (then best-body is full))

(rule (if tastiness is delicate)
      (then best-body is light))

(rule (if tastiness is average)
      (then best-body is light with certainty 30 and
            best-body is medium with certainty 60 and
            best-body is full with certainty 30))

(rule (if tastiness is strong)
      (then best-body is medium with certainty 40 and
            best-body is full with certainty 80))

(rule (if has-sauce is yes and
          sauce is cream)
      (then best-body is medium with certainty 40 and
            best-body is full with certainty 60))

(rule (if preferred-body is full)
      (then best-body is full with certainty 40))

(rule (if preferred-body is medium)
      (then best-body is medium with certainty 40))

(rule (if preferred-body is light)
      (then best-body is light with certainty 40))

(rule (if preferred-body is light and
          best-body is full)
      (then best-body is medium))
```



CLIPS Demonstration

Case Based Reasoning

Case-Based Reasoning

- CBR is based on human information processing (HIP) model in some problem areas
- Law, diagnosis, strategic planning
- Human experts depend heavily on past experiences when solving new problems
- The principle is to find a solution which has been shown to solve problems like your current problem in the past, and adapt it so that it solves the current problem.

What is Case-based Reasoning?

- This has a certain psychological plausibility as a model of what the expert-decision-maker actually does when solving a problem.
- Based on research by Riesbeck & Schank (1989). A good comprehensive description is to be found in Kolodner (1993).
- Three quotes from Roger Schank:
  - "Humans use cases because they don't know what they know - they don't know their own rules - they do things non-reflectively."
  - "The key process in intelligence is the reminding process".
  - "People don't ever reason from first principles. They always choose a matching case. It may be a bad match, but in that case they need more experience."

CBR Components

- A case-based ES consists of
  - a case base
  - a retriever
  - an adapter
  - a refiner
  - an executer
  - an evaluator

Case Base

- A case base functions as a repository of prior cases
- The cases are indexed so that they can be quickly recalled when necessary
- A case contains the general descriptions of old problems

Retriever

- When a new problem is entered into a case based system, a retriever decides on the features similar to the stored cases .
- Retrieval is done by using features of the new cases as indexes into the case base.

Adapter

- An adapter examines the differences between these cases and the current problem
- It then applies rules to modify the old solution to fit the new problem

Refiner

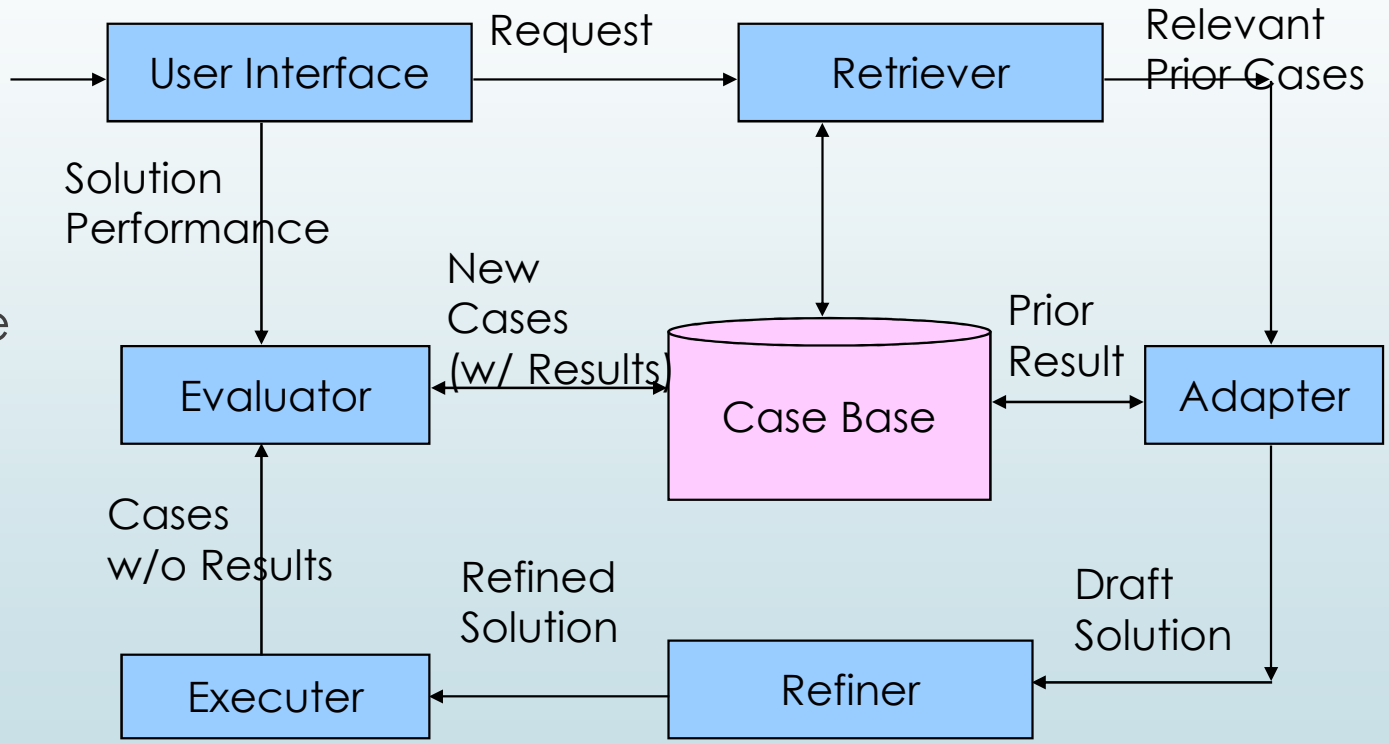
- A refiner critiques the adapted solution against prior outcomes
- One way to do this is to compare it to similar solutions of prior cases.
- If a known failure exists for a derived solution, the system then decides whether the similarities is sufficient to suspect that the new solution will fail

Executer

- Once a solution is critiqued, an executer applies the refined solution to the current problem.

Evaluator

- If the results are as expected, no further analysis is made, and the cases and its solution is stored for use in future problem solving.
- If not, the solution is repaired.



Success Factors

- How much support is provided to end user?
- The ROI achieved
- Expansion of Knowledge Base
- The overall knowledge management

Applications

- Intelligent Information Retrieval
- Case based Searching across WWW
- Technical Support
- Diagnosis
- Planning and Control
- Reasoning

A comparison between rule-based & case-based reasoning

Criterion	Rule-based reasoning	Case-based reasoning
Knowledge unit	Rule	Case
Granularity	Fine	Coarse
Knowledge acquisition	Obtaining rules & hierarchies	Obtaining cases & hierarchies

A comparison between rule-based & case-based reasoning

Criterion	Rule-based reasoning	Case-based reasoning
Explanation mechanism	Backtrace of rules fired	Precedent cases
Characteristic output	Answer + Confidence measure	Answer + precedent cases
Knowledge transfer	Potentially high	Low

A comparison between rule-based & case-based reasoning

Criterion	Rule-based reasoning	Case-based reasoning
Domain requirements	Domain vocabulary, good set of inference rules, rules which hold throughout domain	Domain vocabulary, Case base of example cases, stability: modified cases still hold

Advantages

Rule-based reasoning	Case-based reasoning
Flexible use of knowledge, potentially optimal answers.	Rapid knowledge acquisition, explanation by example

Disadvantages

Rule-based reasoning	Case-based reasoning
Computationally expensive, long development time, impenetrable explanations	Suboptimal solutions, redundancy in knowledge base

Summary

- Brief Introduction to Expert Systems
- Historical Systems – MyCin, Eliza and Dendral
- Rule based system
  - Structure
  - Forward and Backward Chaining
- Case Base Reasoning
  - Structure
  - Comparison to Rule Based System
- **CLIPS Demonstration**