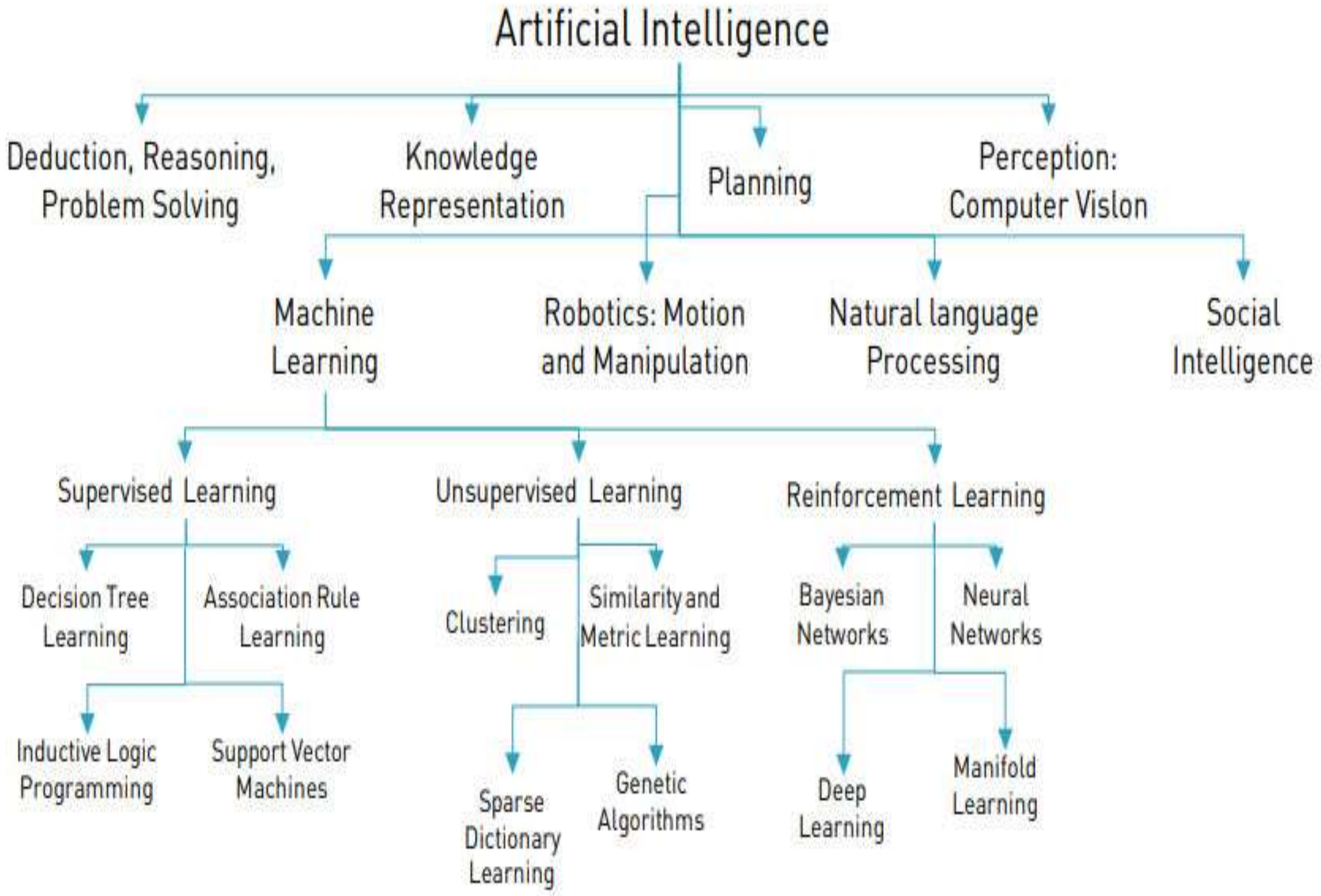
Inductive Learning: Decision Trees, Rule Based Learning, Current Best Hypothesis Search, Least Commitment Search, Neural Network, Reinforcement Learning, Genetic Algorithm

DEDUCTIVE AND INDUCTIVE LEARNING

- **Deductive Learning**
  - Working on already existing facts and knowledge.
  - Deducing new knowledge from knowledge already present in system.
  - Takes the form *if A then B*.
  - Merely an application of knowledge that leads to new knowledge.

- **Inductive learning**
  - Process of learning where a system learns by example.
  - System tries to induce a general rule from a set of observed instances.
  - Uses specific example to reach general conclusion.
  - Classification: task of constructing class definition is called induction or concept learning


Artificial Intelligence
Deduction, Reasoning, Problem Solving
Knowledge Representation
Planning
Perception: Computer Vision
Machine Learning
Robotics: Motion and Manipulation
Natural language Processing
Social Intelligence
Supervised Learning
Unsupervised Learning
Reinforcement Learning
Decision Tree Learning
Association Rule Learning
Inductive Logic Programming
Support Vector Machines
Clustering
Similarity and Metric Learning
Sparse Dictionary Learning
Genetic Algorithms
Bayesian Networks
Neural Networks
Deep Learning
Manifold Learning

Learning

- An agent is learning if
  - It improves its performance on future tasks.
  - Makes observations about the world

- Based on 4 major factors
  - <u>Component</u> to be improved.
  - <u>Prior knowledge</u> of agent.
  - <u>Representations</u> used for data and components.
  - <u>Feedback</u> available to learn from.

Feedback to learn from-Types of Learning

1. **Supervised Learning:**
   - Agent observes some example input output patterns
   - Learns a function that maps input to output
2. **Unsupervised Learning:**
   - No feedback is supplied
   - Agent learns from patterns in input
3. **Reinforcement Learning**:
   - Semi-supervised learning
   - We are given few labeled examples and must make what we can of a large collection of unlabeled examples.

Decision Tree

- Human learn from past experiences or data points. Similarly a machine can be trained to learn from past data points and extract some knowledge.
- Decision tree uses machine learning algorithm to abstract knowledge from data.
- Decision tree is a predictive model that can be viewed as a tree.
- The decision tree is a structure that includes root node, branch and leaf node.
- Each internal node denotes a test on attribute, each branch denotes the outcome of test and each leaf node holds the class label.

- A decision tree represents a function that
  - takes an input a vector of attribute values
  - returns a decision- a single output value.

- Reaches its decision by performing a sequence of tests.

- Expressiveness

  Goal <=> (Path1 V Path2 V …….)

- Decision trees classify the examples by sorting them down the tree from the root to some leaf node, with the leaf node providing the classification to the example.

Decision Tree Algorithm

The **pseudo code** for making decision trees:

1.Create a root node and assign all of the training data to it.

2. Select the best splitting attribute according to criteria.

3.Add a branch to the root node for each value of the split.

4. Split the data into mutually exclusive subsets along the lines of the specific split.

5. Repeat step 2 and 3 for each and every leaf node until a stopping criteria is reached.

There are many algorithms for making decision tree.

Decision tree algorithms differ in three key elements:

1. **Splitting criteria:**

   a) **Which variable to use for first split?** Algorithm uses different measure like information gain, Ginni's Coefficient to compute splitting value.

   b) **How many branches should be allowed for each node?** There could be binary tree or there could be more branches.

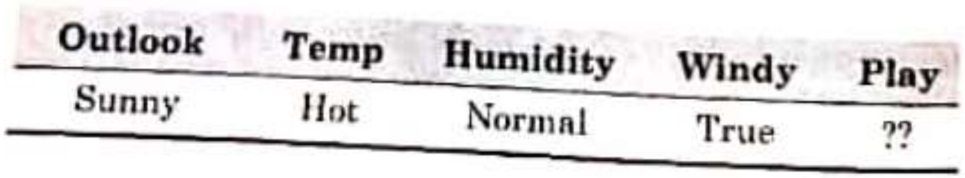
2. **Stopping criteria: When to stop building the decision tree?** There are two ways:

   a) When a certain depth of the branches has been reached and tree becomes more unrendable after that.

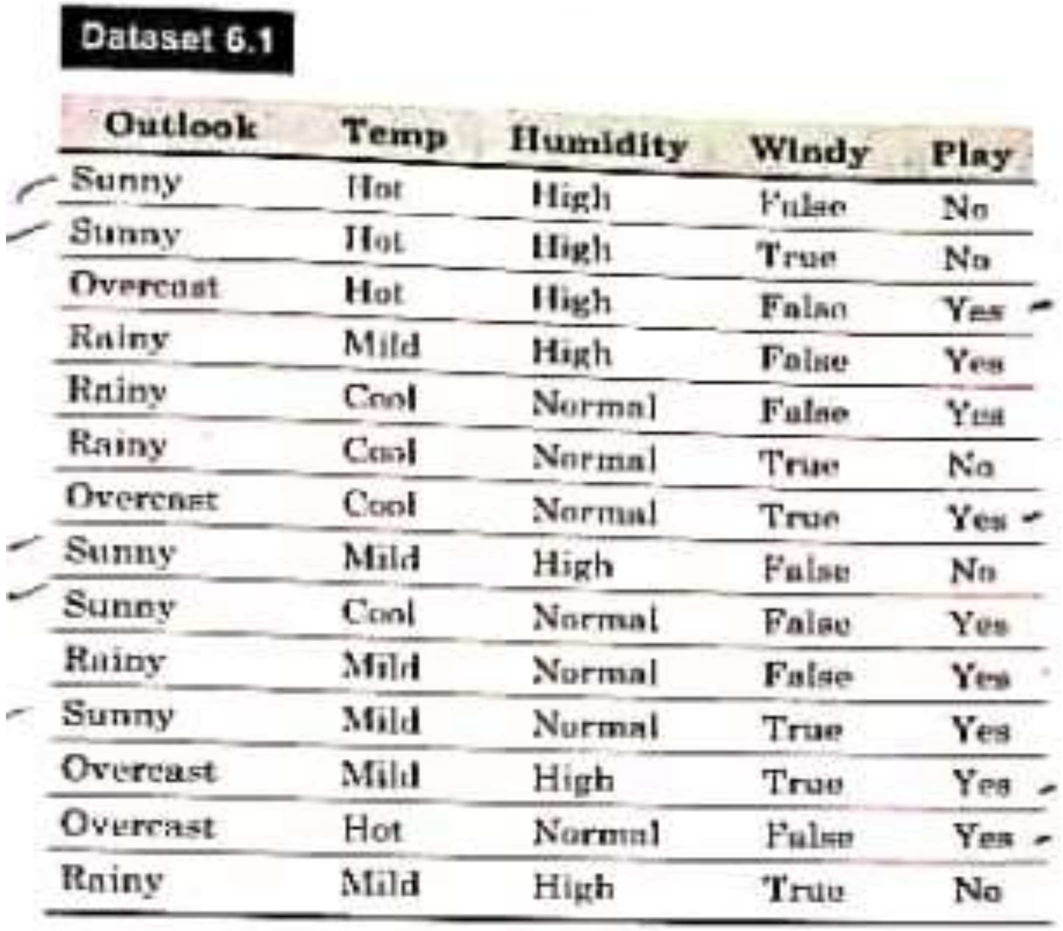
   b) The tree can be stopped when error level at any node is within predefined tolerable levels.

3. **Pruning:** It is the act of reducing the size of decision tree by removing section of the tree that provide little value. The decision tree could be trimmed to make it more balanced and more usable.

Example: Create a decision tree that helps make decision about approving the play of an outdoor game. The objective is to predict the play decision. The decision is- Should the game be allowed or not? Here is the decision problem.



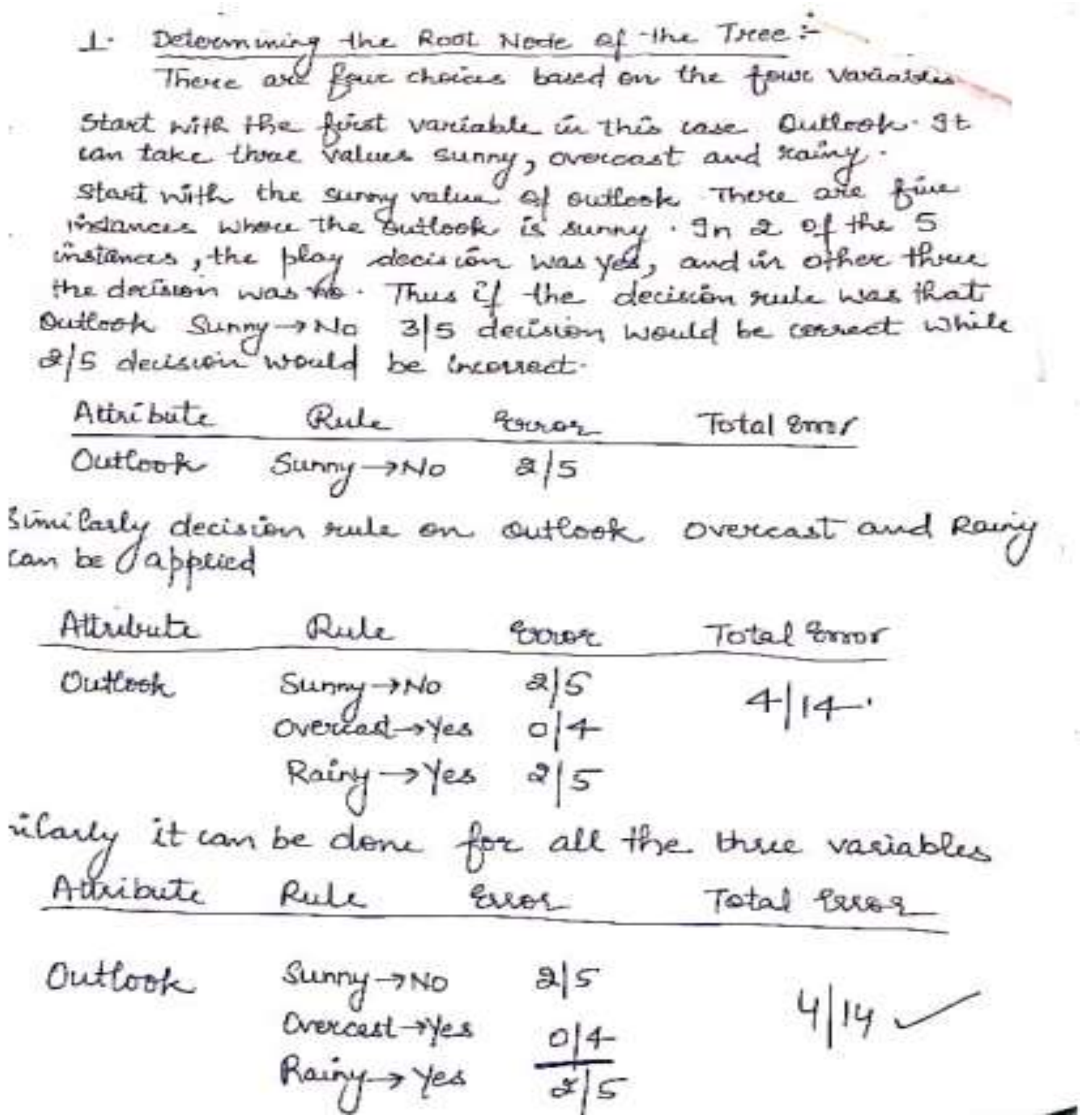
Outlook	Temp	Humidity	Windy	Play
Sunny	Hot	Normal	True	??



Dataset 6.1

Outlook	Temp	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

1. SELECTING THE ROOT



1. Determining the Root Node of the Tree :-

There are four choices based on the four variables

Start with the first variable in this case Outlook. It can take three values sunny, overcast and rainy.

Start with the sunny value of outlook. There are five instances where the outlook is sunny. In 2 of the 5 instances, the play decision was yes, and in other three the decision was No. Thus if the decision rule was that Outlook Sunny → No 3/5 decision would be correct while 2/5 decision would be incorrect.

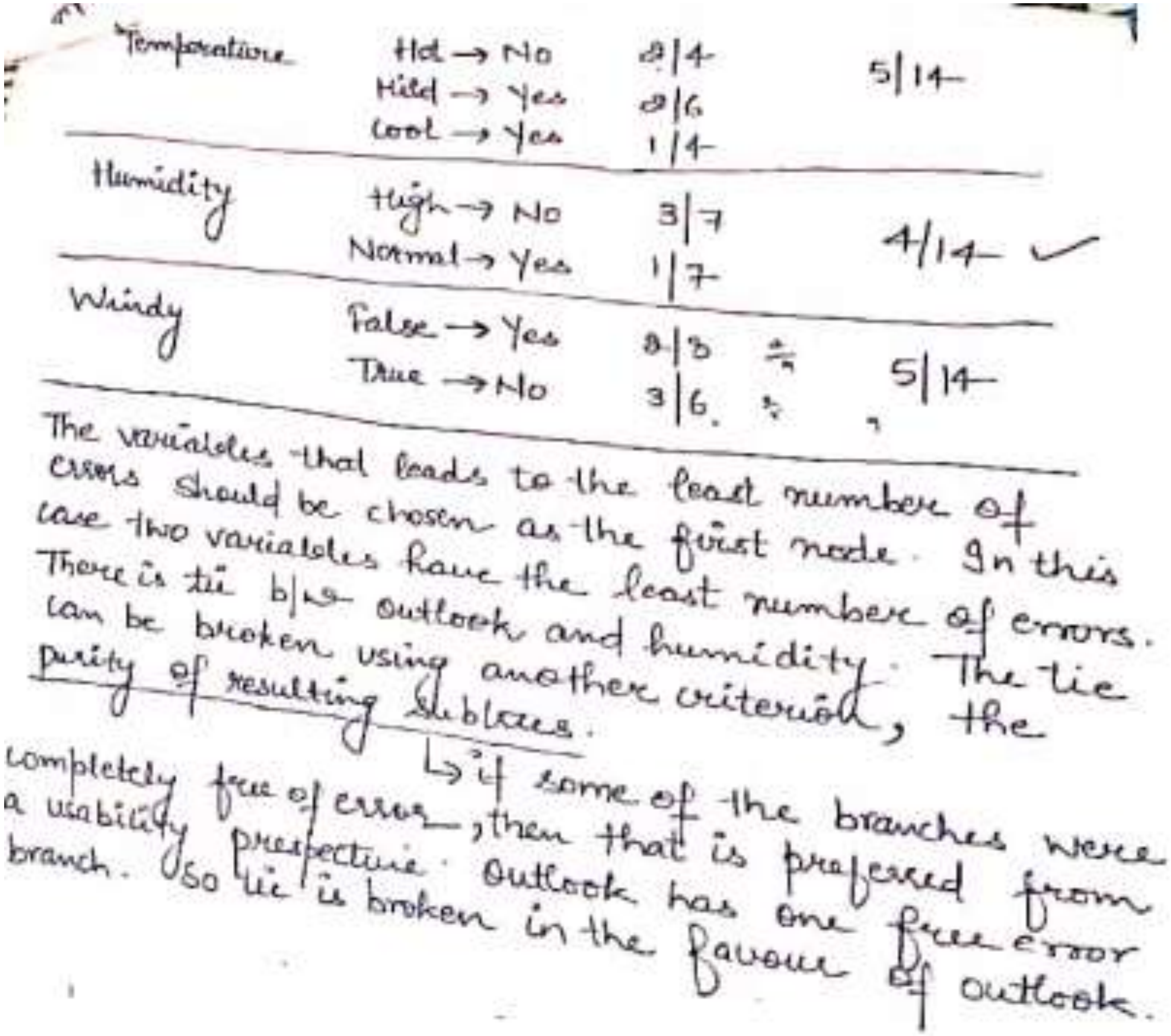
Attribute	Rule	Error	Total Error
Outlook	Sunny → No	2/5	

Similarly decision rule on outlook overcast and Rainy can be applied

Attribute	Rule	Error	Total Error
Outlook	Sunny → No	2/5	4/14
	Overcast → Yes	0/4	
	Rainy → Yes	2/5	

ilarly it can be done for all the three variables

Attribute	Rule	Error	Total Error
Outlook	Sunny → No	2/5	4/14 ✓
	Overcast → Yes	0/4	
	Rainy → Yes	2/5	



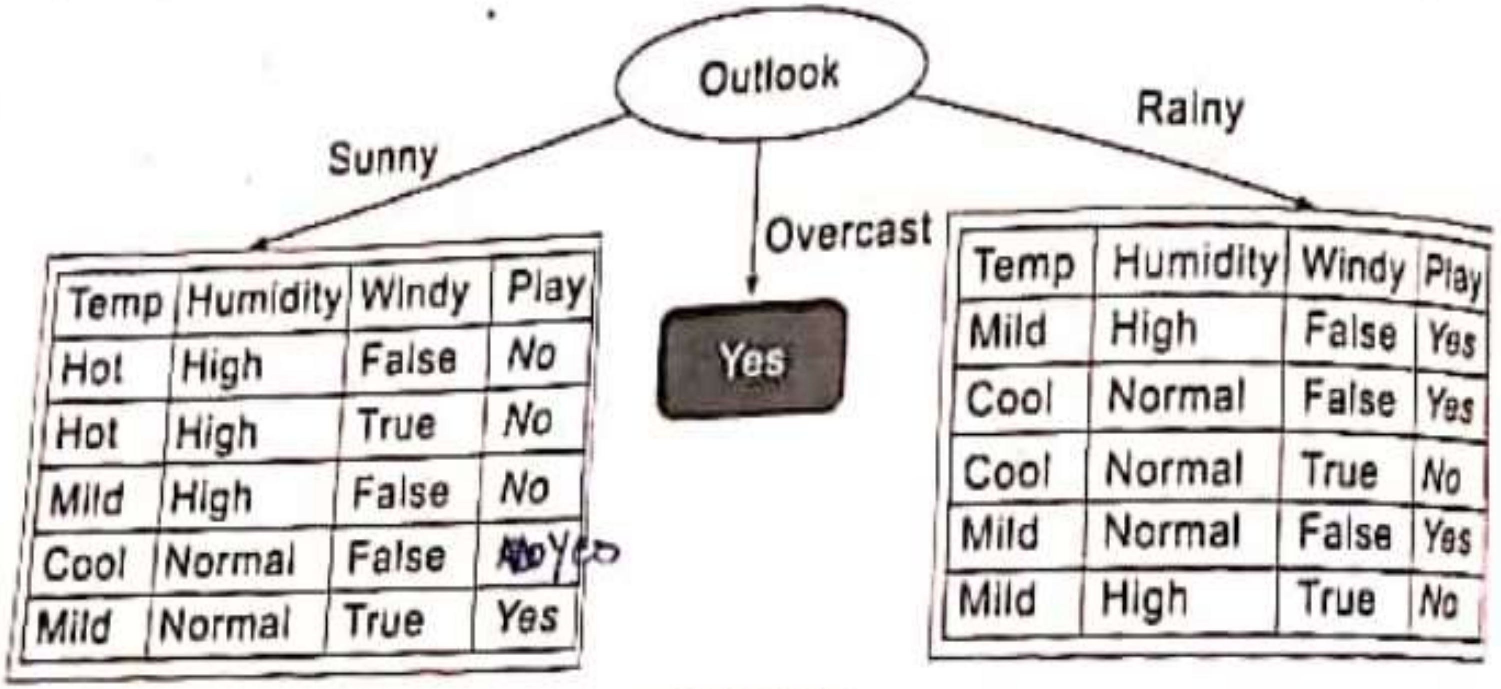
Temperature	Hot → No	2/4	5/14
	Mild → Yes	2/6	
	Cool → Yes	1/4	
Humidity	High → No	3/7	4/14 ✓
	Normal → Yes	1/7	
Windy	False → Yes	2/8	5/14
	True → No	3/6	

The variables that leads to the least number of errors should be chosen as the first node. In this case two variables have the least number of errors. There is tie b/w outlook and humidity. The tie can be broken using another criterion, the purity of resulting subtrees.

↳ if some of the branches were completely free of error, then that is preferred from a usability prespective. Outlook has one free error branch. So tie is broken in the favour of outlook.

2. SPLITTING THE TREE

The decision tree will look like as follows (Figure 6.1) after the first level
splitting.

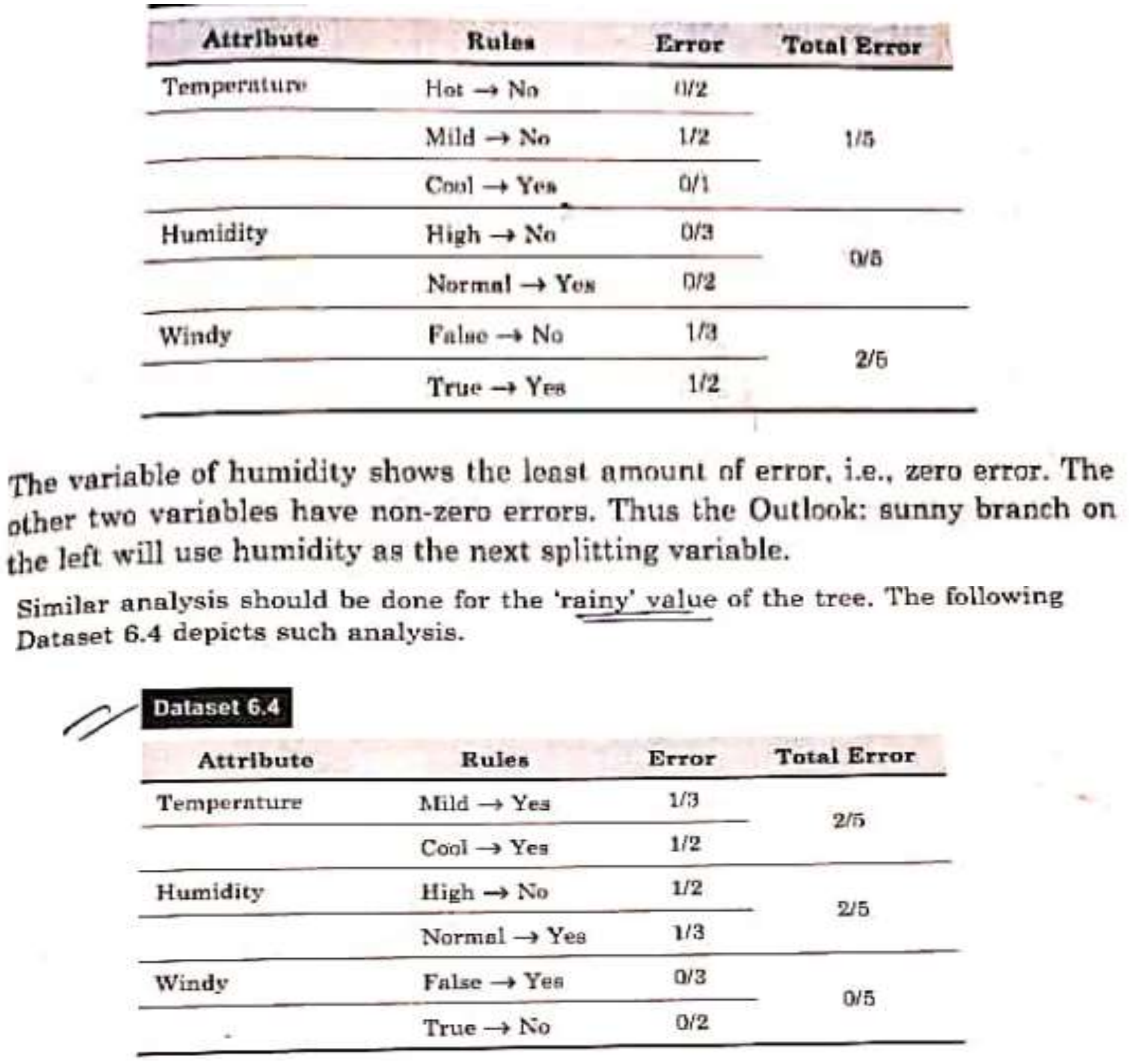


Temp	Humidity	Windy	Play
Hot	High	False	No
Hot	High	True	No
Mild	High	False	No
Cool	Normal	False	~~No~~ Yes
Mild	Normal	True	Yes

Temp	Humidity	Windy	Play
Mild	High	False	Yes
Cool	Normal	False	Yes
Cool	Normal	True	No
Mild	Normal	False	Yes
Mild	High	True	No

FIGURE 6.1

Now determine the next nodes of the tree



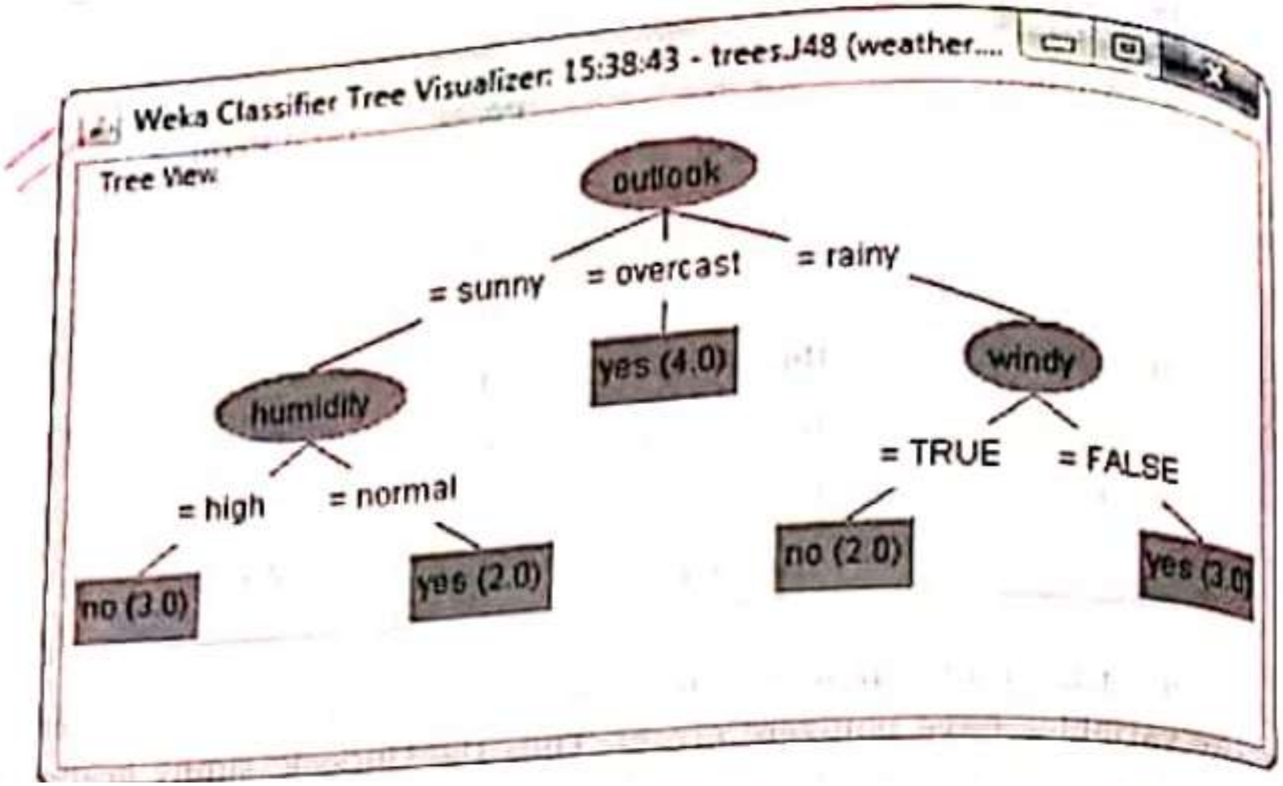
Attribute	Rules	Error	Total Error
Temperature	Hot → No	0/2	1/5
	Mild → No	1/2	
	Cool → Yes	0/1	
Humidity	High → No	0/3	0/5
	Normal → Yes	0/2	
Windy	False → No	1/3	2/5
	True → Yes	1/2	

The variable of humidity shows the least amount of error, i.e., zero error. The other two variables have non-zero errors. Thus the Outlook: sunny branch on the left will use humidity as the next splitting variable.

Similar analysis should be done for the 'rainy' value of the tree. The following Dataset 6.4 depicts such analysis.

Dataset 6.4

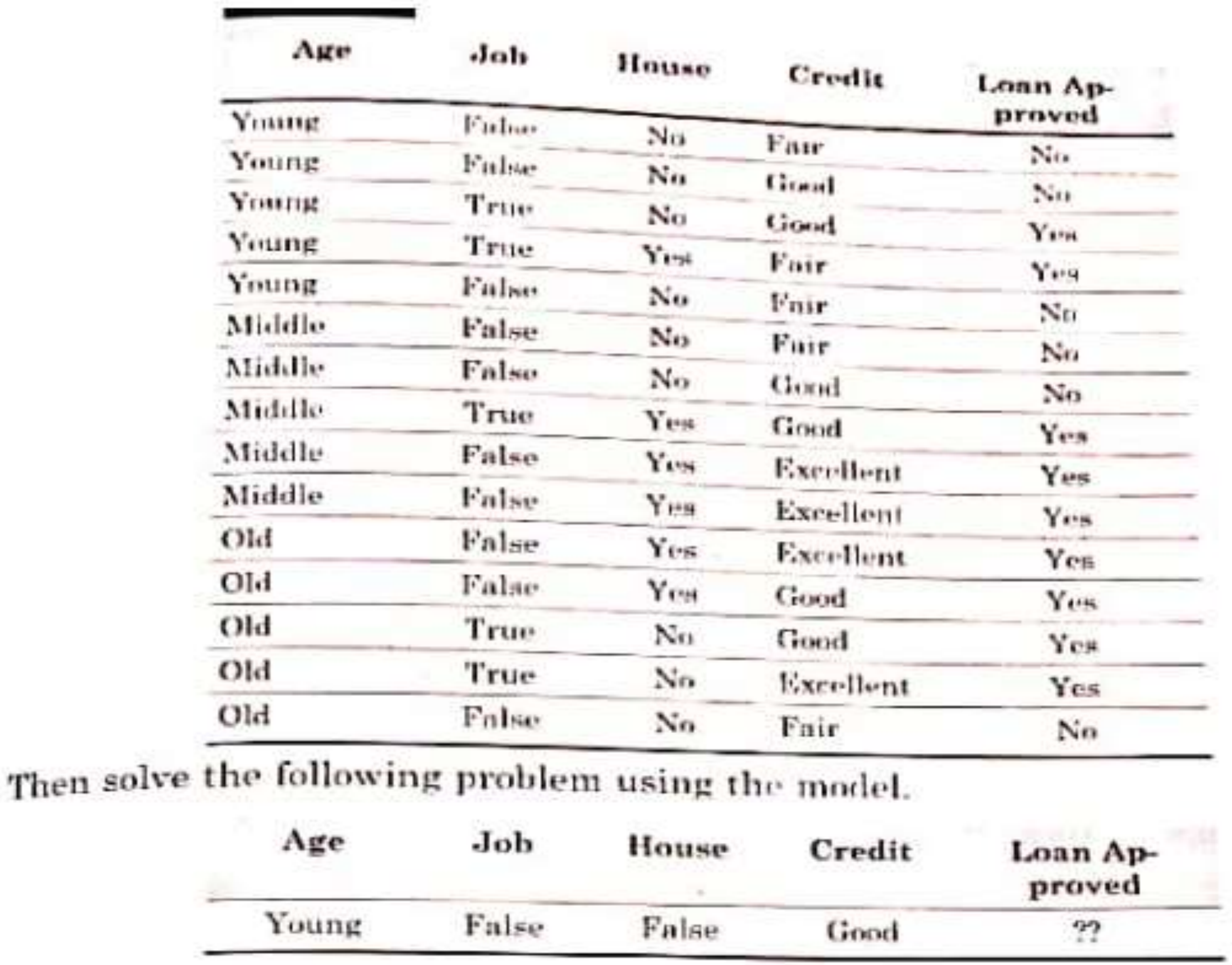
Attribute	Rules	Error	Total Error
Temperature	Mild → Yes	1/3	2/5
	Cool → Yes	1/2	
Humidity	High → No	1/2	2/5
	Normal → Yes	1/3	
Windy	False → Yes	0/3	0/5
	True → No	0/2	



According to the tree, the first question to ask is about outlook. In this problem the outlook is sunny, so the decision problem moves to the 'sunny' branch of the tree. The node in that subtree is humidity. In the problem, humidity is normal. That branch leads to an answer – Yes. Thus, the answer to the play problem is a yes.

Outlook	Temperature	Humidity	Windy	Play
Sunny	Hot	Normal	True	Yes

Example



Age	Job	House	Credit	Loan Approved
Young	False	No	Fair	No
Young	False	No	Good	No
Young	True	No	Good	Yes
Young	True	Yes	Fair	Yes
Young	False	No	Fair	No
Middle	False	No	Fair	No
Middle	False	No	Good	No
Middle	True	Yes	Good	Yes
Middle	False	Yes	Excellent	Yes
Middle	False	Yes	Excellent	Yes
Old	False	Yes	Excellent	Yes
Old	False	Yes	Good	Yes
Old	True	No	Good	Yes
Old	True	No	Excellent	Yes
Old	False	No	Fair	No

Then solve the following problem using the model.

Age	Job	House	Credit	Loan Approved
Young	False	False	Good	??

Advantages and Disadvantages of Decision Trees

- **Advantages:**
  - Easy to use and understand.
  - Can handle both categorical and numerical data.
  - Resistant to outliers, hence require little data preprocessing.
  - New features can be easily added.

- **Disadvantages:**
  - Require some kind of measurement as to how well they are doing.
  - Need to be careful with parameter tuning.

Rule Based Learning

What is Rule Based System?

- A rule-based system is a system that applies human-made rules to store, sort and manipulate data. In doing so, it mimics human intelligence.

- To work, rule-based systems require a set of facts or source of data, and a set of rules for manipulating that data. These rules are sometimes referred to as '**If statements**' as they tend to follow the line of 'IF X happens THEN do Y'.

- Automation software like **Think Automation** is a good example. **It automates processes by breaking them down into steps**. First comes the data or new business event. Then comes the analysis: the part where the system processes the data against its rules. Then comes the automated action.

- **So, what is a rule-based system**?

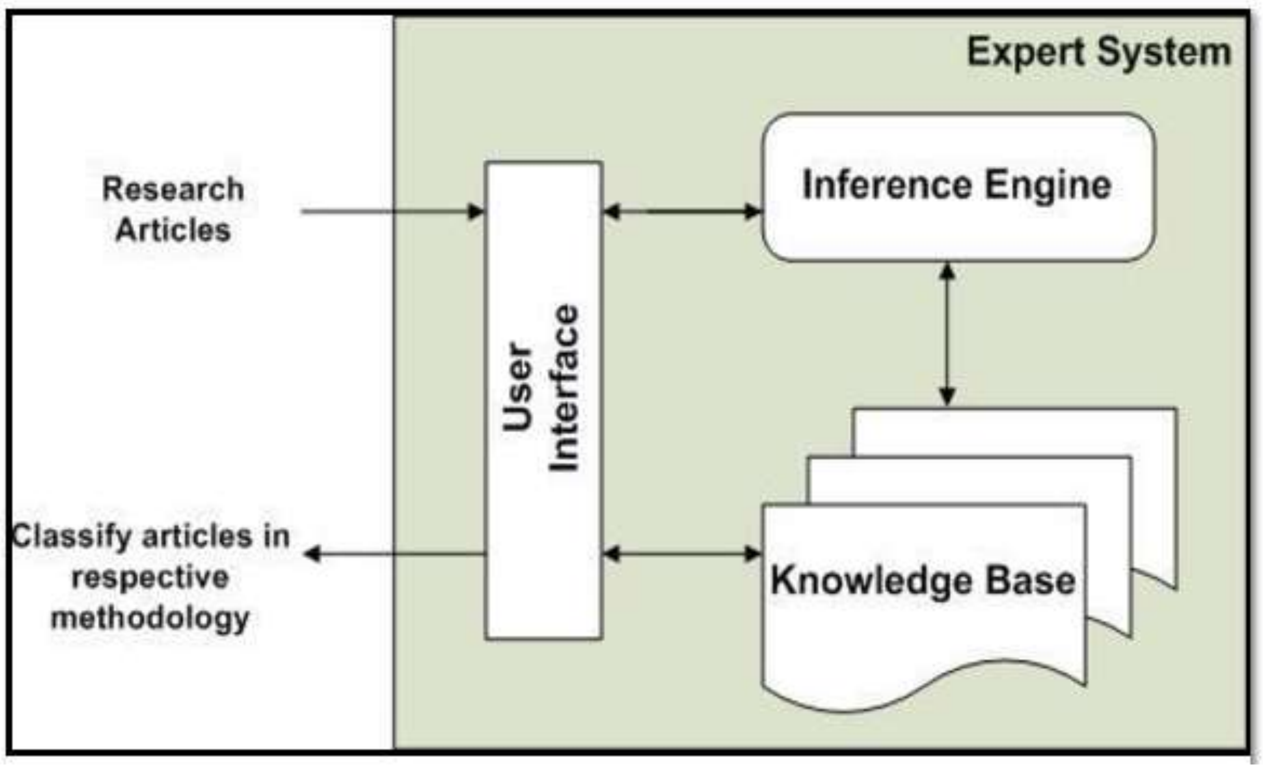
  It’s a logical program that uses pre-defined rules to make deductions and choices to perform automated actions.

How does a Rule Based System work?

- Rule based systems begin with humans programming a set of instructions into a computer.
- Rule based systems are often different from each other in practice, while remaining similar in principle.
- **For example,** a doctor may incorporate the assistance of a computer with a rule based system for aiding the diagnosis process. Alternatively, a rule based system may be applied to game strategy, helping guide a chess player for example.

Every rule based system contains four basic components.

1. Knowledge Base
2. Inference Engine
3. Working Memory
4. User Interface



KNOWLEDGE BASE:

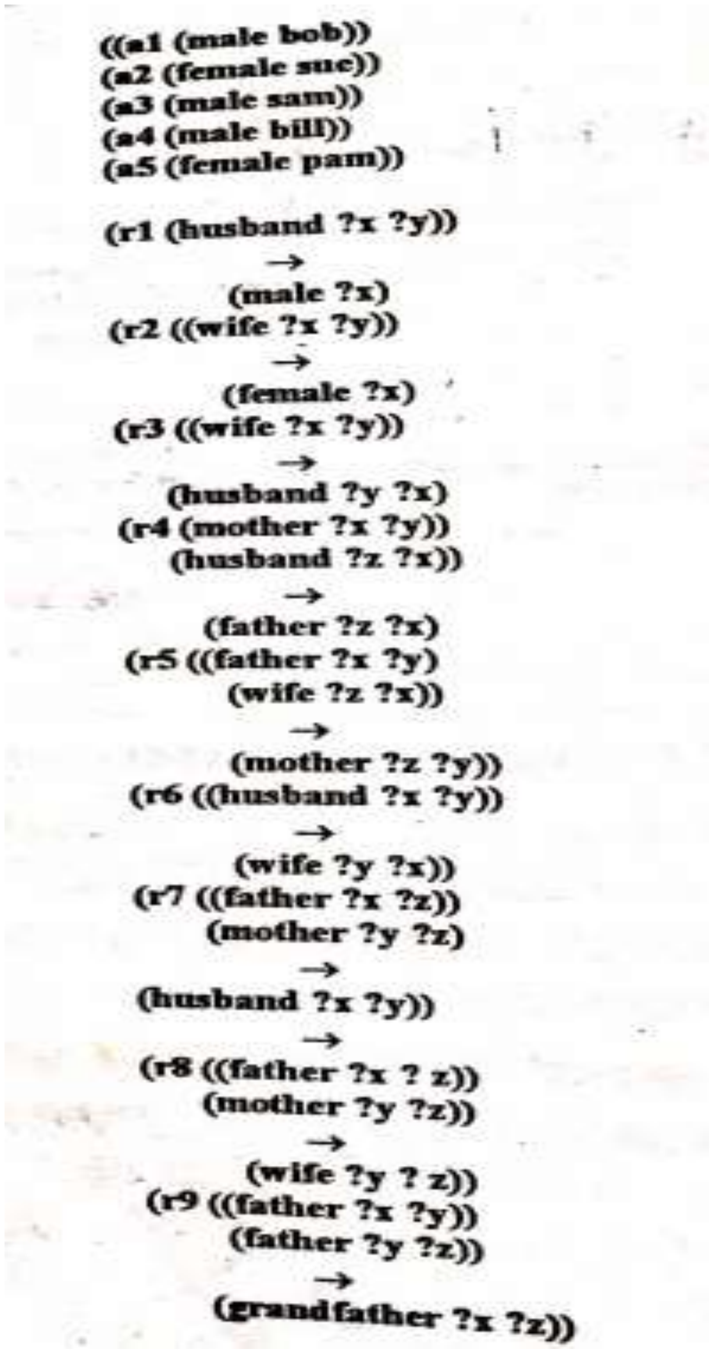
The knowledge base contains facts and rules about some specialized knowledge domain.

Each fact and rule is identified with a name a_1 , a_2 ………., r_1 , r_2 .

For ease left side is separated from the right by the implication symbol $\rightarrow$

Conjuncts on the left are given within parenthesis, and one or more conclusions may follow the implication symbols.

Variables are identified as a symbol preceded by a question mark.



```
((a1 (male bob))
(a2 (female sue))
(a3 (male sam))
(a4 (male bill))
(a5 (female pam))

(r1 (husband ?x ?y))
          →
        (male ?x)
(r2 ((wife ?x ?y))
          →
        (female ?x)
(r3 ((wife ?x ?y))
          →
      (husband ?y ?x)
(r4 (mother ?x ?y))
      (husband ?z ?x))
          →
        (father ?z ?x)
(r5 ((father ?x ?y)
          (wife ?z ?x))
          →
          (mother ?z ?y))
(r6 ((husband ?x ?y))
          →
          (wife ?y ?x))
(r7 ((father ?x ?z))
        (mother ?y ?z)
          →
(husband ?x ?y))
          →
(r8 ((father ?x ? z))
        (mother ?y ?z))
          →
          (wife ?y ? z))
(r9 ((father ?x ?y))
          (father ?y ?z))
          →
        (grandfather ?x ?z))
```

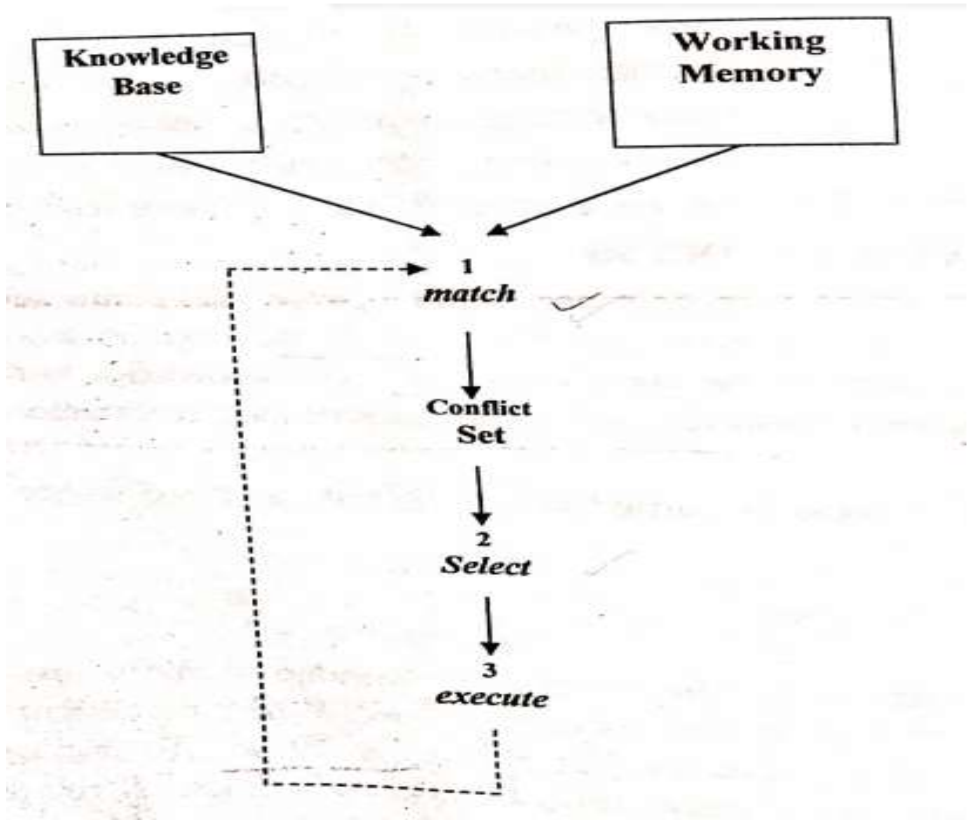
THE INFERENCE PROCESS:

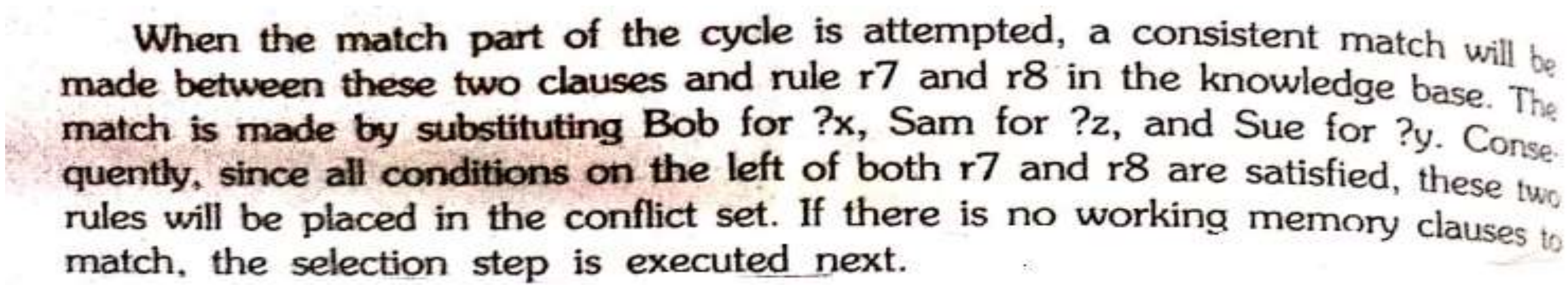
The inference engine accepts user input queries and responses to questions through the I/O interface and uses this dynamic information together with the static knowledge stored in the knowledge base.

The inferring process is carried out recursively in three stages:

1. Match
2. Select
3. Execute

- During the match stage, the contents of working memory are compared to facts and rules contained in the knowledge base.
- When consistent matches are found, the corresponding rules are placed in a conflict set.
- To find an appropriate and consistent match, substitutions may be required.
- Once all the matched rules have been added to the conflict set during a cycle, one of the rules is selected for execution.




When the match part of the cycle is attempted, a consistent match will be made between these two clauses and rule r7 and r8 in the knowledge base. The match is made by substituting Bob for ?x, Sam for ?z, and Sue for ?y. Consequently, since all conditions on the left of both r7 and r8 are satisfied, these two rules will be placed in the conflict set. If there is no working memory clauses to match, the selection step is executed next.

When the left side of sequence of rules is instantiated first and the rule are executed from left to right, the process is called forward chaining.

When the right side of the rules is instantiated first, the left hand conditions become subgoals, this process is called backward chaining.

CURRENT BEST HYPOTHESIS SEARCH

- Hypothesis Space
  - Consistency of Hypothesis with examples.
  - False Negative- says negative but is positive
  - False Positive- says positive but is negative

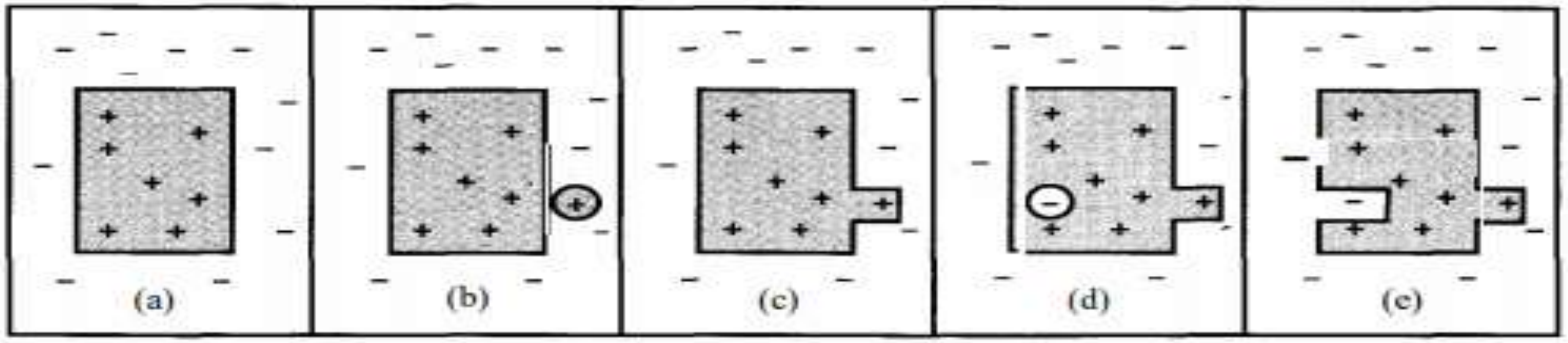
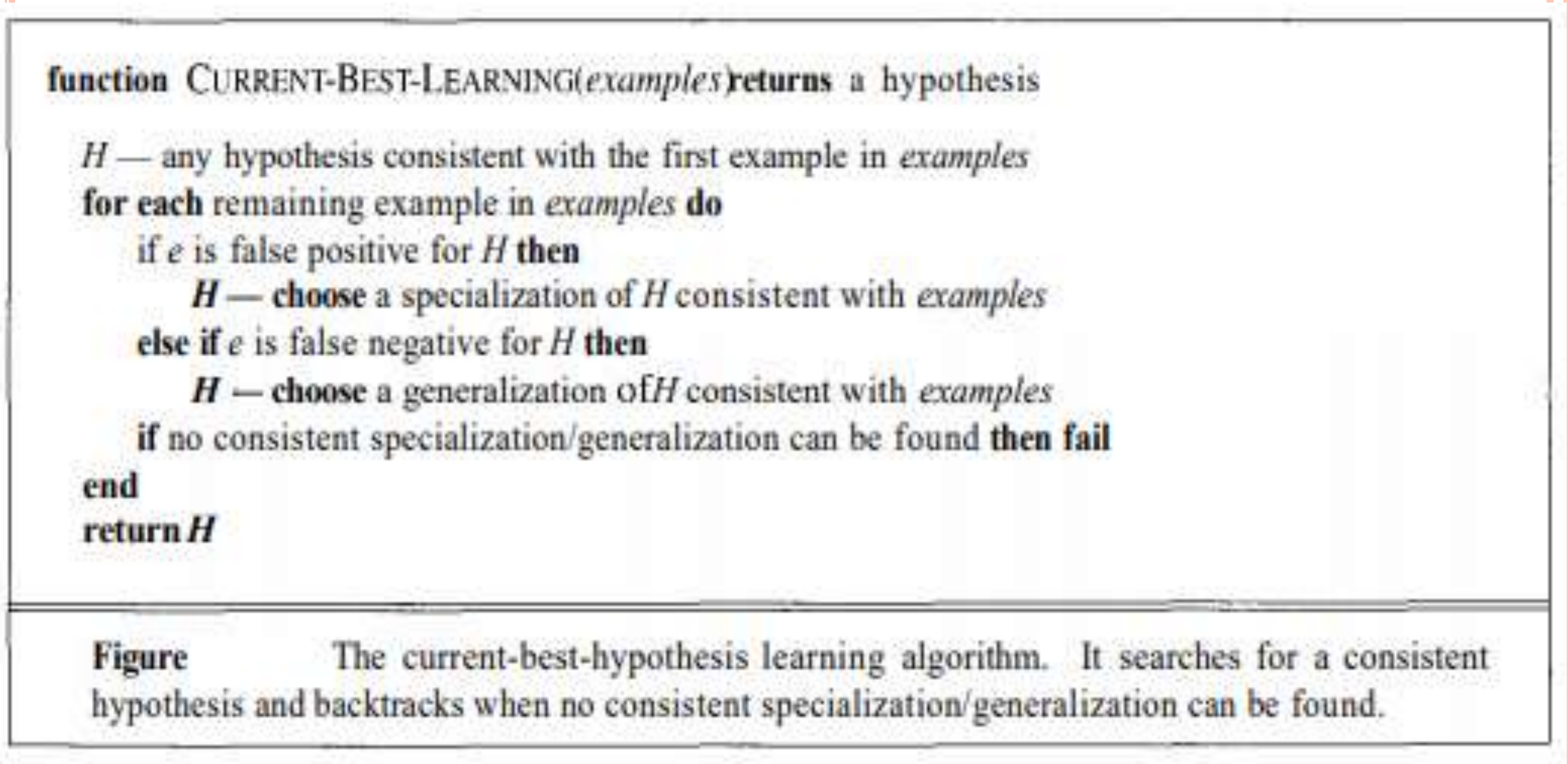


Figure (a) A consistent hypothesis. (b) A false negative. (c) The hypothesis is generalized. (d) A false positive. (e) The hypothesis is specialized.

- Given set of all hypothesis eliminate one that is inconsistent with examples.
  - The set of all hypothesis is infinite.
  - Shrink it to be realizable.

- Current best hypothesis search
  - Maintain a single hypothesis
  - Generalize it for false negative
  - Specialize it for false positive

Algorithm

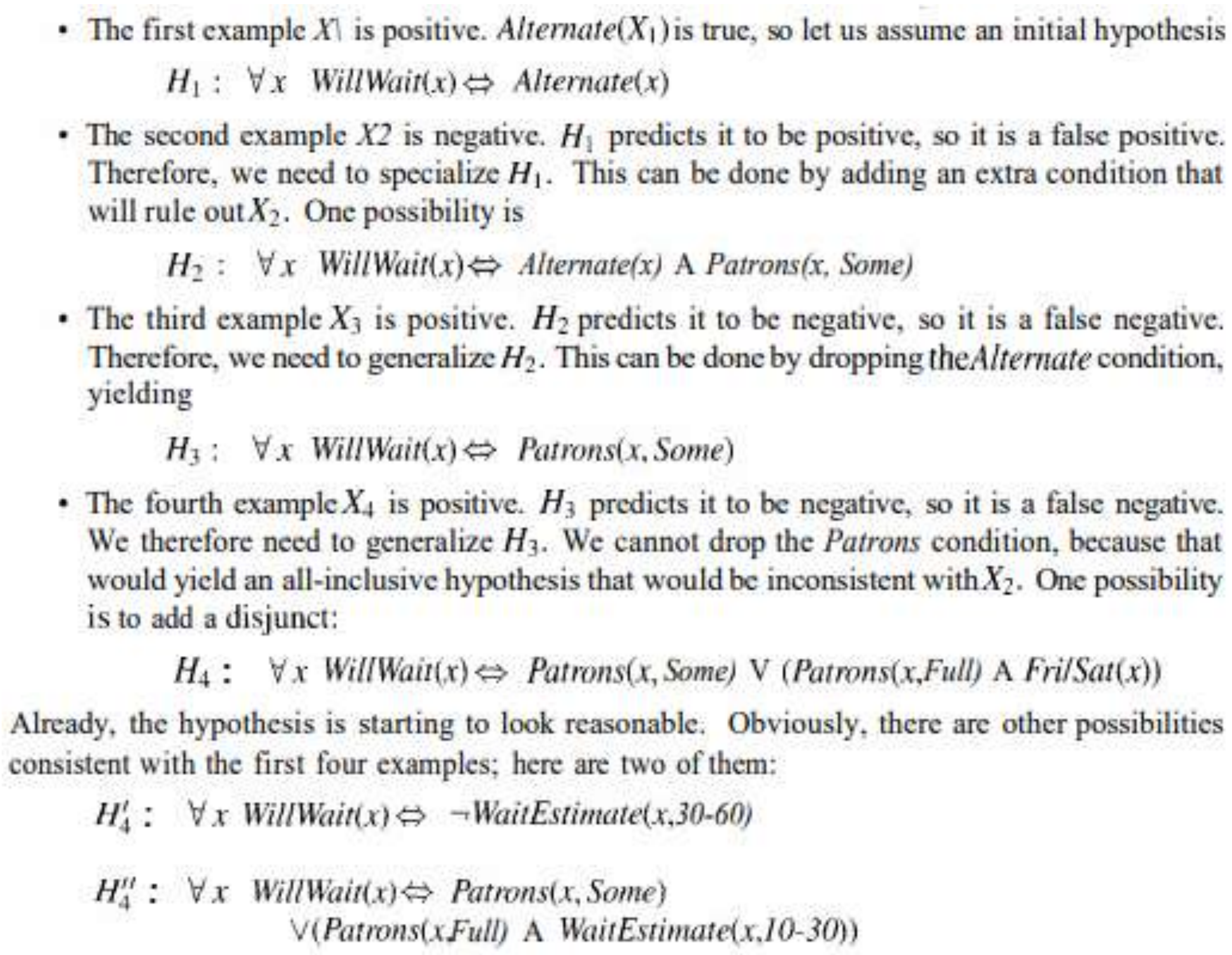


```
function CURRENT-BEST-LEARNING(examples) returns a hypothesis

  H — any hypothesis consistent with the first example in examples
  for each remaining example in examples do
      if e is false positive for H then
          H — choose a specialization of H consistent with examples
      else if e is false negative for H then
          H — choose a generalization of H consistent with examples
      if no consistent specialization/generalization can be found then fail
  end
  return H
```

Figure The current-best-hypothesis learning algorithm. It searches for a consistent hypothesis and backtracks when no consistent specialization/generalization can be found.

Restaurant Example



- The first example X\ is positive. *Alternate*(X_1) is true, so let us assume an initial hypothesis

  H_1 : $\forall x$ *WillWait*(x) $\Leftrightarrow$ *Alternate*(x)

- The second example *X2* is negative. H_1 predicts it to be positive, so it is a false positive. Therefore, we need to specialize H_1. This can be done by adding an extra condition that will rule out X_2. One possibility is

  H_2 : $\forall x$ *WillWait*(x) $\Leftrightarrow$ *Alternate(x)* $\wedge$ *Patrons(x, Some)*

- The third example X_3 is positive. H_2 predicts it to be negative, so it is a false negative. Therefore, we need to generalize H_2. This can be done by dropping the *Alternate* condition, yielding

  H_3 : $\forall x$ *WillWait*(x) $\Leftrightarrow$ *Patrons*(x, *Some*)

- The fourth example X_4 is positive. H_3 predicts it to be negative, so it is a false negative. We therefore need to generalize H_3. We cannot drop the *Patrons* condition, because that would yield an all-inclusive hypothesis that would be inconsistent with X_2. One possibility is to add a disjunct:

  H_4 : $\forall x$ *WillWait*(x) $\Leftrightarrow$ *Patrons*(x, *Some*) $\vee$ (*Patrons*(x,*Full*) $\wedge$ *Fri/Sat*(x))

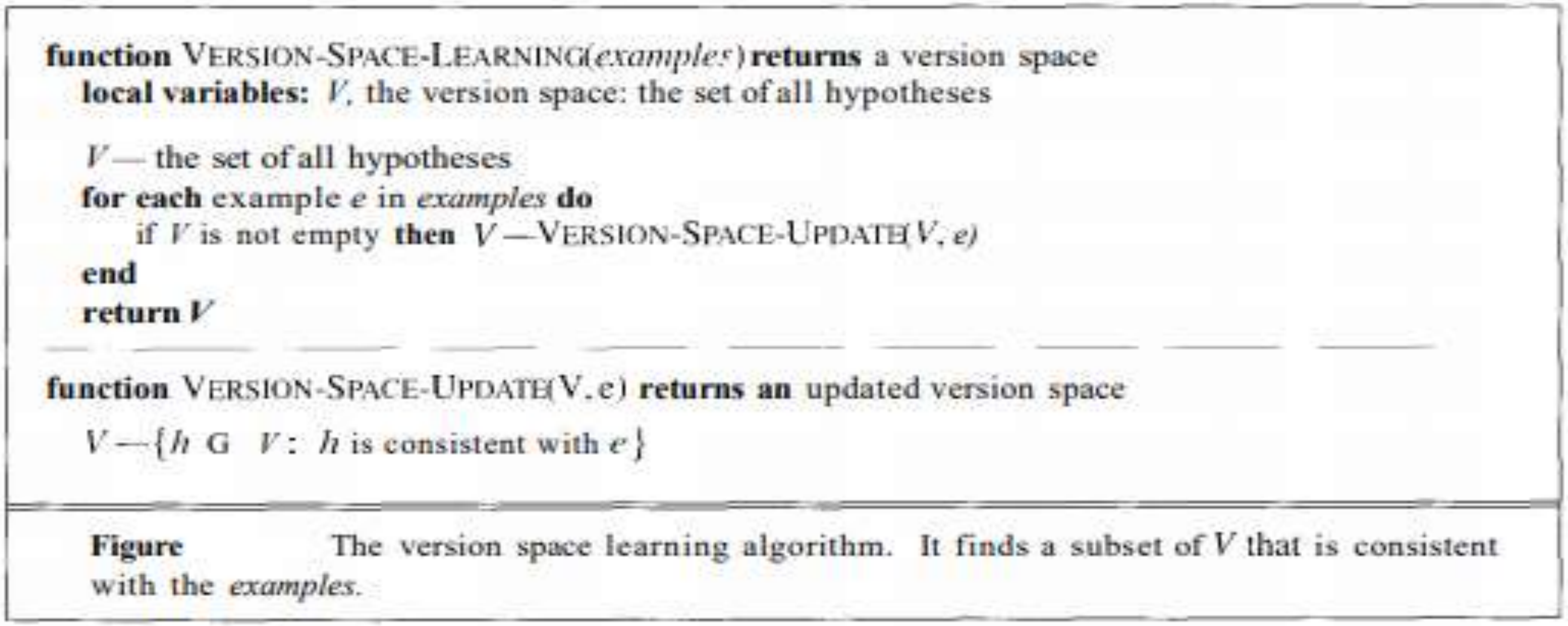
Already, the hypothesis is starting to look reasonable. Obviously, there are other possibilities consistent with the first four examples; here are two of them:

H'_4 : $\forall x$ *WillWait*(x) $\Leftrightarrow$ $\neg$*WaitEstimate*(x,30-60)

H''_4 : $\forall x$ *WillWait*(x) $\Leftrightarrow$ *Patrons*(x, *Some*)
$\vee$(*Patrons*(x,*Full*) $\wedge$ *WaitEstimate*(x,10-30))

LEAST COMMITMENT SEARCH

- Incremental Approach such that consistency is guaranteed without backtracking
- Version Space: set of hypothesis remaining



```
function VERSION-SPACE-LEARNING(examples) returns a version space
   local variables: V, the version space: the set of all hypotheses

   V — the set of all hypotheses
   for each example e in examples do
      if V is not empty then V —VERSION-SPACE-UPDATE(V, e)
   end
   return V

function VERSION-SPACE-UPDATE(V, e) returns an updated version space
   V —{h G V: h is consistent with e}
```

Figure The version space learning algorithm. It finds a subset of *V* that is consistent with the *examples*.

- Incremental Approach
- Least commitment algo because it makes no arbitrary choice.
- Partial Ordering on Hypothesis space i.e. each boundary will not be a point but a set of hypotheses called a boundary set.
  - Generalization/specialization
  - G-set: more generalized boundary
  - S-set: more specialized boundary

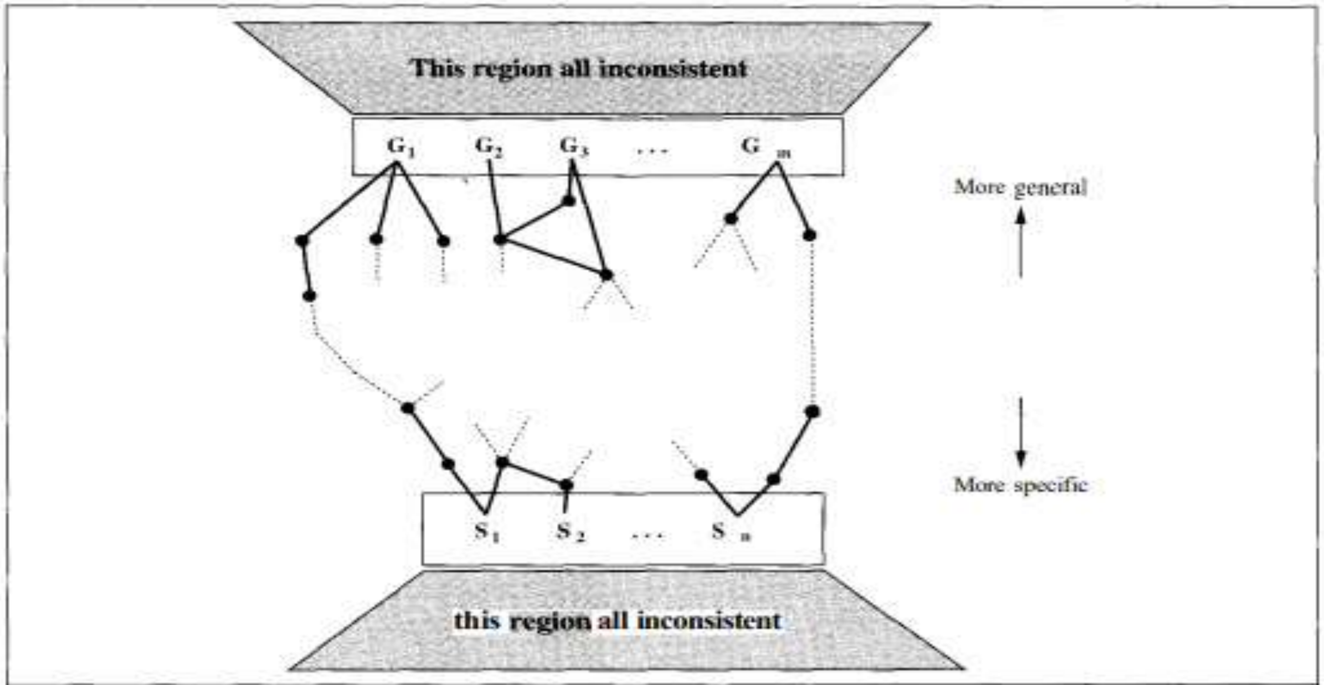
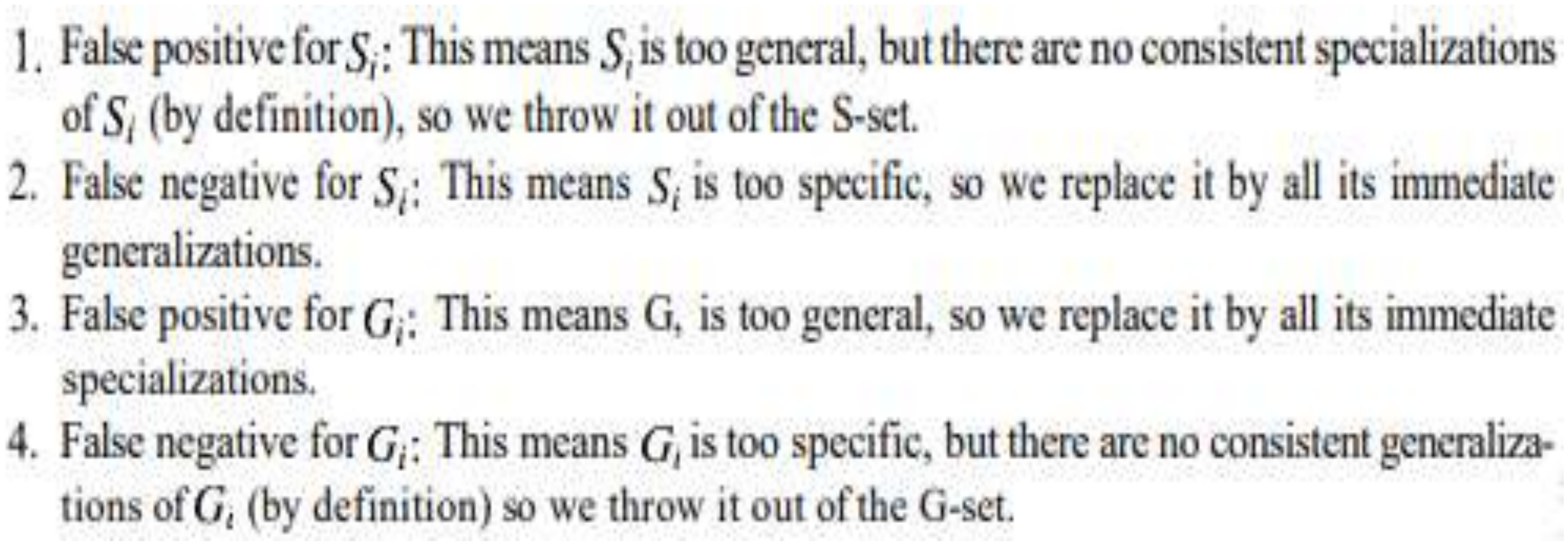

This region all inconsistent
G_1 G_2 G_3 ... G_m
More general
More specific
S_1 S_2 ... S_n
this region all inconsistent

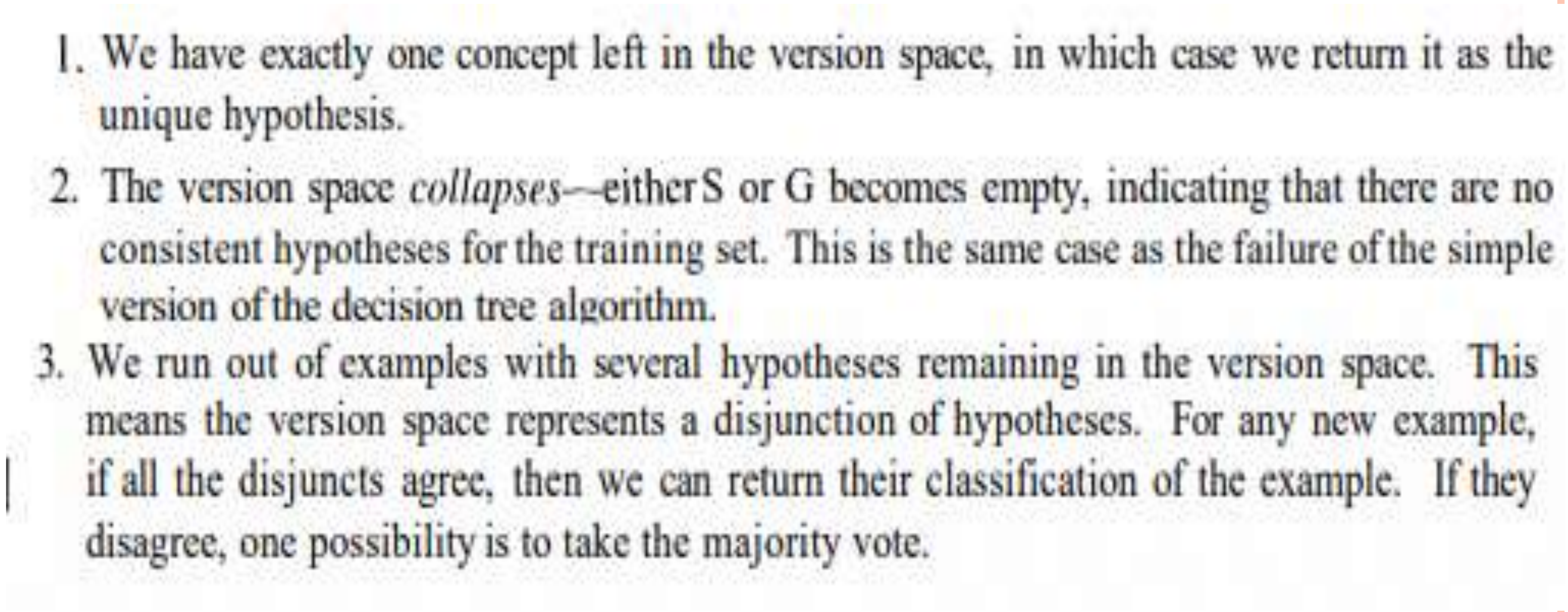
Figure The version space contains all hypotheses consistent with the examples.

- **The new instance may be false negative or false positive.**
- **For**



1. False positive for S_i: This means S_i is too general, but there are no consistent specializations of S_i (by definition), so we throw it out of the S-set.
2. False negative for S_i: This means S_i is too specific, so we replace it by all its immediate generalizations.
3. False positive for G_i: This means G, is too general, so we replace it by all its immediate specializations.
4. False negative for G_i: This means G_i is too specific, but there are no consistent generalizations of G_i (by definition) so we throw it out of the G-set.

- **Continue these operations for each new instance until one of the following happens.**



1. We have exactly one concept left in the version space, in which case we return it as the unique hypothesis.
2. The version space *collapses*—either S or G becomes empty, indicating that there are no consistent hypotheses for the training set. This is the same case as the failure of the simple version of the decision tree algorithm.
3. We run out of examples with several hypotheses remaining in the version space. This means the version space represents a disjunction of hypotheses. For any new example, if all the disjuncts agree, then we can return their classification of the example. If they disagree, one possibility is to take the majority vote.