Journal of Innovative Technology and Education, Vol. 13, 2026, no. 1, 1 - 13
HIKARI Ltd, www.m-hikari.com
https://doi.org/10.12988/jite.2026.1167

Teaching Decision Trees for Learning Expert System Programming

Ping-Tsai Chung

School of Engineering, Computer Science and AI
Long Island University
1 University Plaza
Brooklyn, New York 11201, USA

This article is distributed under the Creative Commons by-nc-nd Attribution License.
Copyright © 2026 Hikari Ltd.

Abstract

Expert systems are built to solve a wide range of problems in domains such as medicine, science, engineering, manufacturing, education, and many other fields. A Rule-Based Expert System has the following components: *the knowledge base, the inference engine, and a user interface* that communicates with users. A Decision Tree is a hierarchical data structure consisting of *nodes*, which store information or knowledge, and *edges*, which connect the nodes. In this paper, we discuss that *the inference* or *reasoning process* of Expert System can be easier to identify by a decision tree structure through a forward reasoning inference or a backward reasoning inference. Forward reasoning inference is data-driven; its reasons for facts to find out what solutions (or conclusions) follow from the facts. However, backward reasoning inference is goal-driven, which needs to find facts that support the hypothesis. A small animal classification example will be discussed.

For learning expert system programming effectively, we introduce two software tools and discuss their advantages. Firstly, CLIPS (C Language Integrated Production System) is discussed. CLIPS was designed using C programming language at the NASA/Johnson Space Center with the specific purpose of providing high portability, low cost, and easy integration with external systems. Secondly, an interactive logic programming tool, Visirule is discussed; VisiRule was offered by LPA, a small, private UK company specializing in practical Artificial Intelligence. Finally, we propose a *Rubric* for Outcome Assessments for learning expert system programming with decision Trees for anyone who is interested in practical expert system programming.

Keywords: Decision trees, inference, reasoning, forward chaining, backward chaining, expert system programming, CLIPS, and VisiRule

I. INTRODUCTION

A Decision Tree is a convenient model of algorithms involving comparisons in which: internal nodes represent comparisons and leaves (outcomes) (i.e., nodes without outgoing edges) represent outcomes. Let's examine a small animal classification example in Figure 1, without loss of generality, the internal node represents a comparison (a decision) and a leaf represents (an outcome). In figure 2, we represent the Decision Tree in Figure 1 into a flowchart. We then discuss a reasoning inference that can be derived from the Decision Tree in a forward reasoning inference or backward reasoning inference in Section 2.

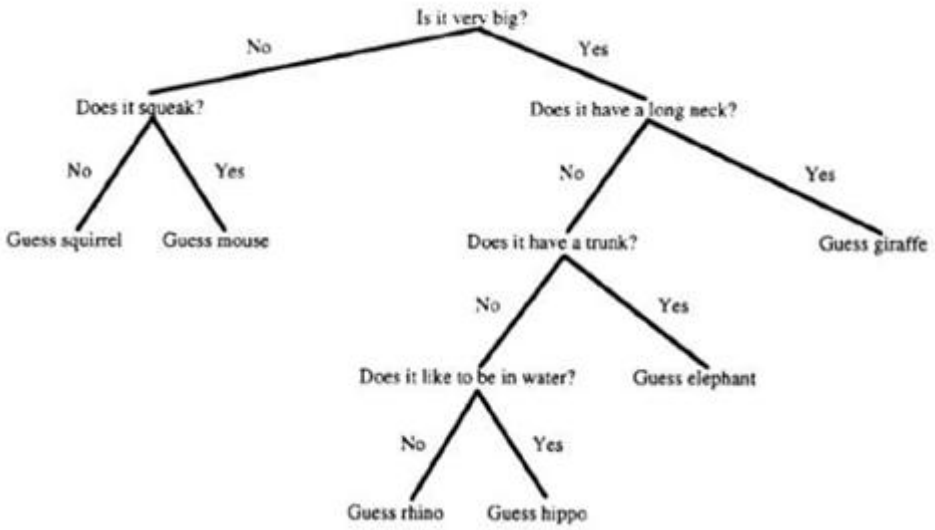

Figure 1. A Decision Tree for Animal Classification.

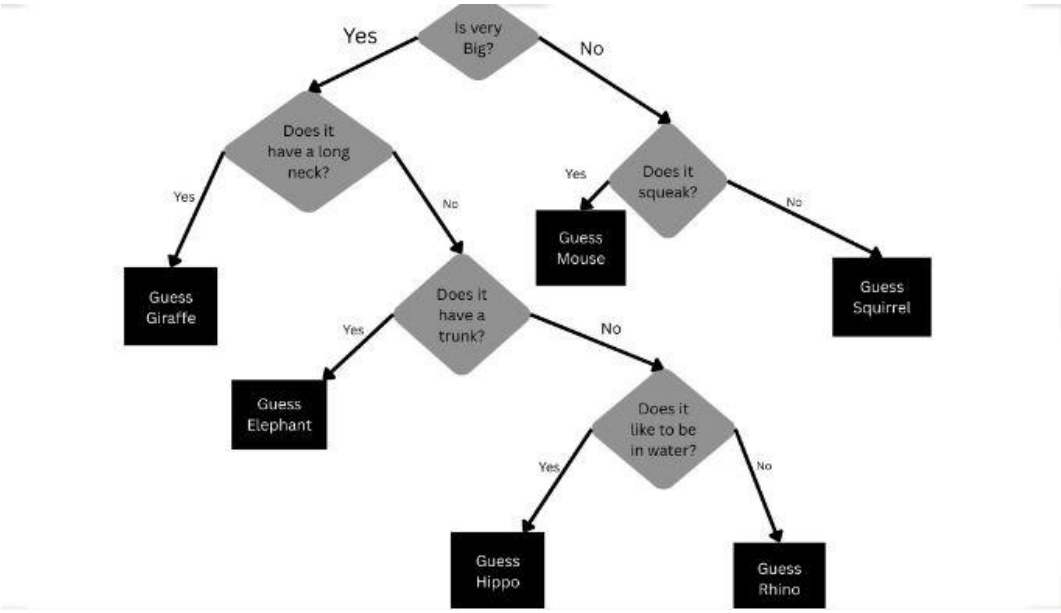

Figure 2. A Flow Chart for Decision Tree for Animal Classification.

Note that there is a unique path from the starting point (the root) of a Decision tree to an outcome, We can show that any comparison-based sorting algorithm can be represented by a decision tree with number of leaves (outcomes) ≥ $n!$, the Height of binary tree with $n!$ leaves ≥ $\lceil \log_2 n! \rceil$, and Minimum number of comparisons in the worst case ≥ $\lceil \log_2 n! \rceil$ for any comparison-based sorting algorithm $\lceil \log_2 n! \rceil \approx n \log_2 n$ [1].

II. FORWARD CHAINING VS BACKWARD CHAINING

In this section, we discuss an inference that can be derived from the Decision Tree. A **forward reasoning inference** or **forward chaining**, is a data-driven AI process that starts with known facts, applies inference rules, and logically derives to build knowledge until a goal is reached, moving forward from causes to effects. A backward reasoning inference or backward chaining is a goal-driven AI process that starts with the goal and works backward to determine which facts must be true to achieve that goal.

Follow the Forward chaining rules (i.e., Rule 1 to Rule 6), derived from the Decision Tree in Figure 1 are listed below.

Rule1: If (it is very Big = No) AND (it squeaks = No) Then (it is a Squirrel).
Rule 2: If (it is very Big = No) AND (it squeaks = Yes) Then (it is a Mouse).

Rule 3: If (it is very Big = Yes) AND (it has a long neck = No) AND (it has a trunk = No) AND (it likes water = NO) Then (it is a Rhino).

Rule 4: If (it is very Big = Yes) AND (it has a long neck = No) AND (it has a trunk = No) AND (it likes water = Yes) Then (it is a Hippo).

Rule 5: If (it is very Big = Yes) AND (it has a long neck = No) AND (it has a trunk = Yes) Then (it is an Elephant).

Rule 6: If (it is very Big = Yes) AND (it has a long neck = Yes) Then (it is a Giraffe).

Table 1. Inference Rules for Forward Chaining for the Small Animal Classification.

Backward chaining cases (i.e. Rule 7 to Rule 12), derived from the Decision Tree in Figure 1 are listed below.

Rule 7: If (it is a Squirrel) then (it squeaks = No) AND (it is very big = No).

Rule 8: If (it is a Mouse) then (it squeaks = Yes) AND (it is very big = No).

Rule 9: If (it is a Rhino) then (it likes water= No) AND (it has a trunk = No) AND (it has a long neck = No) AND (it is very big = Yes).

Rule 10: If (it is a Hippo) then then (it likes Water = Yes) AND (it has a trunk = No) AND (it has a long neck = No) AND (it is very big = Yes).

Rule 11: If (it is an Elephant) then (it has a trunk = Yes) AND (it has a long neck = No) AND (it is very big = Yes).

Rule 12: If (it is a Giraffe) then (it has a long neck = Yes) AND (it is very big = Yes).

Table 2. Inference Rules for Backward Chaining for the Small Animal Classification.

III. EXPERT SYSTEM PROGRAMMING

For learning expert system programming effectively, we introduce two software tools: CLIPS and VisiRule and discuss their advantages.

CLIPS (C Language Integrated Production System) was designed using C programming language at the NASA/Johnson Space Center with the specific purpose of providing high portability, low cost, and easy integration with external systems. We could follow Rule 1 to Rule 6 to implement CLIPS code as follows:

```
/* The CliPS program running by the Initial-fact */

(defrule start-up
        (initial-fact)
        =>
        (printout t " ************************************************************" crlf)
        (printout t " * Animal Classification Expert System                      *" crlf)
        (printout t " ************************************************************" crlf)
        (printout t " " crlf)
        (printout t " " crlf)
        (printout t " " crlf)
        (printout t " Now, We begin our consulting" crlf))
/* Advising User to enter Yes or No to update the facts*/
(defrule first
        (initial-fact)
        =>
        (printout t " Enter yes or no to answer the question" crlf)
        (printout t " " crlf)
        (printout t " Let us start" crlf)
        (printout t " " crlf)
        (printout t "Is the animal very big?" crlf)
        (bind ?answer1 (read))
        (assert (animal-big ?answer1)))

/* Repeat advising User to enter Yes or No to update the facts through a forward reasoning
inference or forward chaining process */

(defrule checking_animal_yes
        (animal-big yes)
        =>
        (printout t "Does it have a long neck?" crlf)
        (bind ?answer2 (read))
        (assert (neck-long ?answer2)))

(defrule neck_long_yes
        (neck-long yes)
        =>
        (printout t "The animal is: Guess giraffe" crlf))

(defrule neck_long_no
        (animal-big yes)
        (neck-long yes)
        =>
        (printout t "Does it have a trunk?" crlf)
        (bind ?answer3 (read))
        (assert (having-trunk?answer3)))
```

```
(defrule having_trunk_yes
        (having-trunk yes)
        =>
        (printout t "The animal is: Guess elephant" crlf))

(defrule having_trunk_no
        (animal-big yes)
        (neck-long no)
        =>
        (printout t "Does it like to be in water?" crlf)
        (bind ?answer4 (read))
        (assert (like-water?answer4)))

(defrule like_water_yes
        (like-water yes)
        =>
        (printout t "The animal is: Guess hippo" crlf))

(defrule like_water_no
        (animal-big yes)
        (neck-long no)
        (having-trunk no)
        =>
        (printout t "The animal is: Guess rhino" crlf))

(defrule checking_animal_no
        =>
        (printout t "Does it squeak?" crlf)
        (bind ?answer5 (read))
        (assert (does-squeak?answer5)))

(defrule does_squeak_yes
        (does-squeak yes)
        =>
        (printout t "The animal is: Guess mouse" crlf))

(defrule does_squeak_no
        (animal-big no)
        =>
        (printout t "The animal is: Guess squirrel" crlf))

(defrule try_again
        (initial-fact)
          =>
        (printout t "Try again? (y/n)" crlf)
        (assert (response (read))))
/* Do another test */
(defrule do-another
?if <- (initial-fact)
?r <- (response yes)
        =>
```

```
        (retract ?if ?r)
        (assert (initial-fact)))

(defrule do-not-do-another
?r <- (response ~yes)
        =>
        (retract ?r))
```

Table 3. CLIPS Code for the Small Animal Classification.

Now we discuss an interactive logic programming tool, VisiRule offered by LPA**,** a small, private UK company specializing in practical Artificial Intelligence. VisiRule is a versatile and powerful visual software tool which enables business professionals to create expert systems using expert knowledge. VisiRule helps you draw your decision logic and will build the interactive questions for you from the VisiRule diagram. As answers are given, VisiRule executes the logic associated with that part of the Decision Tree flowchart and asks the next question. Upon completion, VisiRule returns the conclusions reached and any recommendations.

Follow the Forward chaining rules (i.e. Rule 1 to Rule 6), derived from the Decision Tree in Figure 1, a VisiRule diagram is drawn as follows. VisiRule provides interactive questions to User's answer. Finally, VisiRule returns the advice.

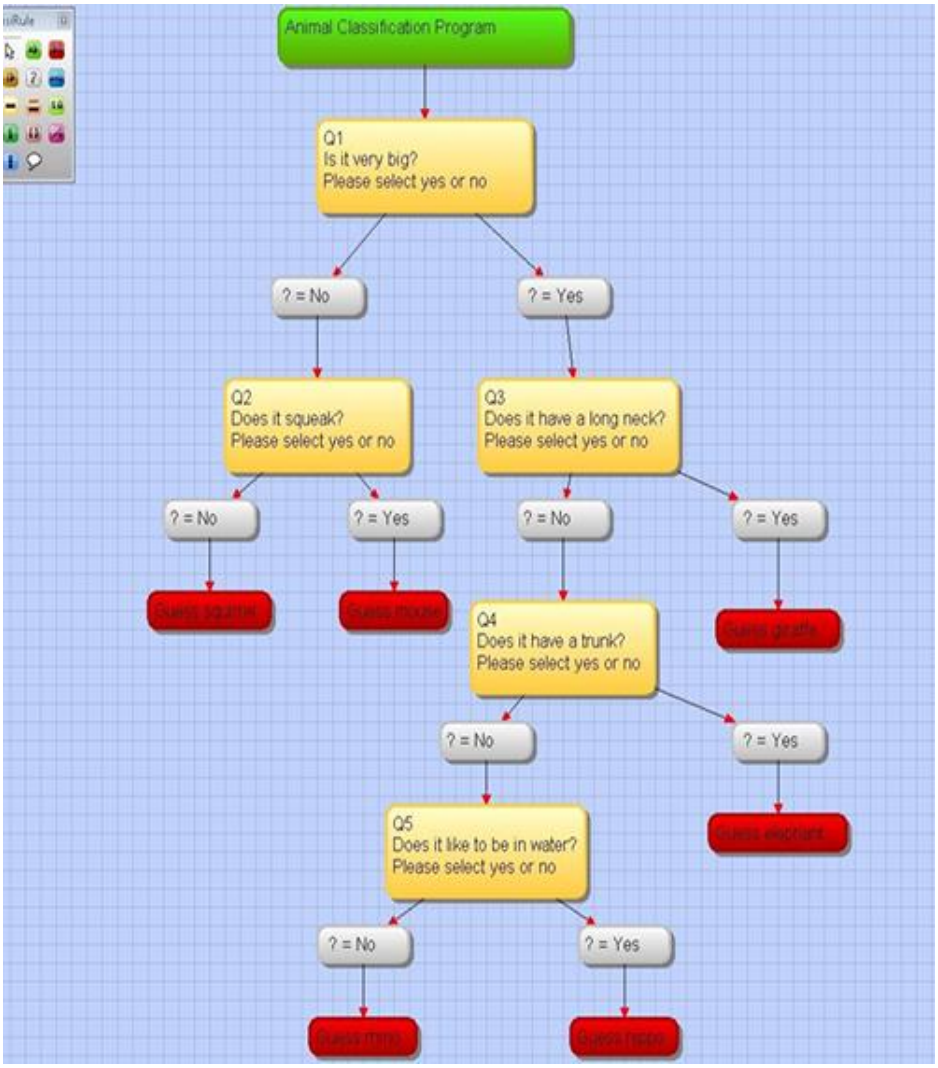

Figure 3. A VisiRule Diagram for the Small Animal Classification.

IV. KNOWLEDGE BASE AND DECISION TREES

A complete animal classification example in [2] contained twenty-one Inference Rules (i.e., Rule-001 to Rule-021) as follows.

Rule-001:	IF (the animal is not very big AND (it squeaks) THEN it is a *Mouse*
Rule-002:	IF (the animal is not very big) AND (it does not squeak) THEN it is a *Squirrel*
Rule-003:	IF (the animal is very big) AND it (has a long neck) THEN it is a *Giraffe*
Rule-004:	IF (the animal is very big) AND (it does not have a long neck) AND (it has a trunk) THEN it is an *Elephant*
Rule-005:	IF (the animal is very big) AND (it does not have a long neck) AND (it does not have a trunk) AND (it likes to be in water) it is a *Hippo*
Rule-006:	IF (the animal is very big) AND (it does not have a long neck) AND (it does not have a trunk) AND (it does not like to be in water) THEN it is a *Rhino*
Rule-007:	IF (the animal has hair) THEN it is a *Mammal*
Rule-008:	IF (the animal gives milk) THEN it is a *Mammal*
Rule-009:	IF (the animal has feathers) THEN it is a *Bird*
Rule-010:	IF (the animal flies) AND (It lays eggs) THEN it is a *Bird*
Rule-011:	IF (the animal is a mammal) AND (It eats meat) THEN it is a *Carnivore*
Rule-012:	IF (the animal is a mammal) AND (It has pointed teeth) AND (It has claws) AND (Its eyes point forward) THEN it is a *Carnivore*
Rule-013:	IF (the animal is a mammal) AND (It has hoofs) THEN it is an *Ungulate*

Rule-014:	IF (the animal is a mammal) AND (It chews cud) THEN it is an *Ungulate* AND (It is even toed)
Rule-015:	IF (the animal is a carnivore) AND (It has a tawny color) AND (It has dark spots) THEN it is a *Cheetah*
Rule-016:	IF (the animal is a carnivore) AND (It has a tawny color) AND (It has black strips) THEN it is a *Tiger*
Rule-017:	IF (the animal is an ungulate) AND (It has long legs) AND (It has long neck) AND (It has a tawny color) AND (It has dark spots) THEN it is a *Giraffe*
Rule-018:	IF (the animal is an ungulate) AND (It has a white color) AND (It has black strips) THEN it is a *Zebra*
Rule-019:	IF (the animal is a bird) AND (It does not fly) AND (It has long legs) AND (It has a long neck) AND (It is black and white) THEN it is an *Ostrich*
Rule-020:	IF (the animal is a bird) AND (It does not fly) AND (It swims) AND (It is black and white) THEN it is a *Penguin*
Rule-021:	IF (the animal is a bird) AND (It is a good flyer) THEN it is an *Albatross*

Table 4. Inference Rules for the Complete Animal Classification in [2].

Remark: We could follow the Principle of Section III to develop a decision flow chart for Decision Tree and implement an expert system in CLIPS (or in VisiRule). Due to paper space, we omit this part.

A *Rubric* for Outcome Assessments is given below for instructors to assess students' learning expert system programming with Decision Tree. This rubric can be used for assessing learners from K-12, or college students with any background who are interested in learning practical Artificial Intelligence technologies.

Rating Code	**4**	**3**	**2**	**1**
Description of Competence	**Excellent**	**Satisfactory**	**Average**	**Inadequate**
Understanding of the Problem	• Shows excellent understanding of the Problem. Shows 100% documentation	• Shows good understanding of the Problem. Shows 90% documentation	• Shows good understanding of the Problem. Shows 80% documentation	• Shows good understanding of the Problem. Shows less than 70% documentation
Forward Reasoning Inference	• Shows 100% Correct reasoning process	• Shows most of Correct (i.e., 80%) reasoning process • Results	• Shows some Correct (i.e., 60%) reasoning process Results	• Less than 60% reasoning process Results
Backward Reasoning Inference	• Shows useful reasoning Results	• Shows some useful reasoning Results	• Shows few reasoning Results	• Shows insufficient reasoning Results
The rigorous analysis of their efficiency	• The program is running effectively	• The program is running, but the Decision Tree needs to be modified slightly.	• The program is running, but the Decision Tree needs to be modified.	• The program is not running, effectively due to the Decision Tree being not correct.

Table 5. A *Rubric* for Outcome Assessments for learning expert system programming.

Acknowledgements. I would like to express my sincere thanks to my students who completed my courses taught in Long Island University, Brooklyn Campus, New York from 2009 to 2018: CS 666, Introduction of Artificial Intelligence (a graduate course) and course: CS 162 Introduction of Artificial Intelligence (an undergraduate course).

IV. CONCLUSIONS

We discussed that *the inference* or *reasoning process* of Expert System can be easier to identify by a Decision Tree structure through a forward reasoning inference or a backward reasoning inference. For learning expert system programming effectively, we introduced two software tools and discussed their advantages. CLIPS was designed using C programming language at the NASA/Johnson Space Center with the specific purpose of providing high portability, low cost, and easy integration with external systems. VisiRule is a no-code/low-code visual authoring package used to build rule-based expert systems and automate decision-making processes. Recently, a new version of VisiRule employs AI to mine and induces decision trees directly from historical data sets. For assessing the students' learning of expert system programming, we provided a *Rubric* for Outcome Assessments for anyone who is interested in practical expert system programming.

References

[1] A. Levitin, *The Design & Analysis of Algorithms*, Addison Wesley, 2007.

[2] B. Avron and E. Feigenbaum, *The Handbook of artificial intelligence*, volume 2, 1982.

[3] J. Giarratano and G. Riley, *Expert Systems – Principles and Programming*, PWS Publishing Company, 1998.

[4] R. A. Akerkar and P. S. Sajja, *Knowledge-Based Systems*, Jones and Bartlett Publishers, 2010.

[5] P. H. Winston, *Artificial Intelligence*, Addison-Wesley Publishing Company, 1984.

[6] G. F. Luger, *Artificial Intelligence – Structures and Strategies for Complex Problem Solving*, Pearson Education, Inc., 2009.

[7] P. T. Chung, Discovery of Rules for Web Intelligence - A Case Study Based on Rough Sets and Bayes' Theorem", International Journal of Digital Content Technology and Its Applications (JDCTA), *Advanced Institute of Convergence IT* (AICIT), **10** (2016), 45-56.

[8] P. T. Chung, B. X. Chen, A Knowledge-Based Decision System for Healthcare Diagnosis and Advisory, *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, (2011), 2357-2362. https://doi.org/10.1109/icsmc.2011.6084030

[9] C. Mavropoulos, P. T. Chung, A Rule-based Expert System: Speakeasy – Smart Drink Dispenser, *The Proceedings of IEEE Long Island System Applications, and Technology*, (2014).

Received: January 5, 2026; Published: January 19, 2026