Dog Obedience Advisor Expert System

Final Report

John New

1 January 2007

johnn198@gmail.com

http://www.beaglebytes.com/

Table of Contents

Abstract

This report discusses the development of a Dog Obedience Advisor expert system.

The Advisor has two parts: a Dog Obedience Competition Advisor; and a Dog Obedience Breed Advisor. The former is designed to check a handler and dog for eligibility and success in relation to various classes of canine Obedience competitions; the latter to provide obedience information about selected dog breeds contained in a sample database on which users can perform various types of searches.

The report summarises the original proposal for the development of a general Dog Competition Checker expert system. It explains the reasons for revising the project and building a specific Dog Obedience Advisor instead. It provides a general overview and detailed description of the Dog Obedience Competition Advisor and Dog Obedience Breed Advisor. It discusses some of the problems and issues encountered during the development of the Dog Obedience Advisor. Some final observations are made about the expert system.

Introduction

This section summarises the original project proposal to build a general **Dog Competition Checker** expert system and explains the reasons for revising the proposal and building a specific **Dog Obedience Advisor** instead.

Summary of the Original Project Proposal

The original proposal for this project was to build a Dog Competition Checker expert system.

The intention was that this expert system would check a handler and dog for eligibility and success in relation to various types of canine competitions held under the auspices of the Royal NSW Canine Council, Australia [21], and endorsed by the Australian National Kennel Council [2], the peak administrative body for canine affairs in Australia. It was thought that an expert system would conceivably be very useful for this knowledge domain because of the variety of canine competitions and the many rules that handlers and dogs must follow to succeed in a competition.

Canine competitions include such as Agility, Earthdog, Endurance, Flyagility, Flyball, Herding, Jumping, Lure Coursing, Obedience, Showing, and Tracking. Many of these competitions have different levels (or classes). Each class, ranging from novice to expert, has multiple tasks. Judging is tough with points swiftly deducted for the slightest fault. Disqualification is not unusual.

The expert system was to provide a quick and easy way for a handler to check: first, whether a dog was eligible to compete in a certain type and class of competition; and, second, whether a dog had successfully completed the series of tasks required and was, therefore, eligible for an award or title endorsed by the Australian National Kennel Council. The expert system was to provide some tips for improving performance at competition and avoiding breaches of the rules.

The intention was that the expert system would include rules for checking Obedience competitions. This is partly because canine obedience is a prerequisite for success at any other type of canine competition, and partly because the rules for Obedience are fairly complex. It was also intended that the expert system would include rules for other types of canine competitions, if it was possible to include them in the time available.

Revisions to the Original Project Proposal

After starting work on the expert system, and after building the rules to check Obedience competitions, it became clear that it would be advantageous to revise the original proposal. After confirmation [11], work continued on a revised project that:

- Narrowed the focus of the domain from canine competitions in general to Obedience competitions in particular
- Broadened the treatment of Obedience by providing Obedience information about various dog breeds

There were three main reasons for revising the proposal [13].

First, the majority of rules used to check various classes of Obedience competitions are IF ... THEN rules and the expert system would use the same rule structure to check other types of competitions. Although the expert system uses some complex checking to work out the final result (`Passed` or `Failed`) for each class, it did not seem sensible to add other types of canine competitions that used the same relatively simple and predictable rule structure.

Second, there are numerous types of canine competitions in addition to Obedience. As work progressed, it became apparent that it would be impossible to add all other types of canine competitions in the time available to finish the system. Therefore, it would become necessary to pick and choose. But which types of canine competition should be chosen? Every type is popular. Those that are, perhaps, not quite as popular (for example, Endurance, which is extremely demanding for both handler and dog) nevertheless have very enthusiastic supporters. After discussing the problem with some Instructors at this student's local Dog Training Club, which trains for a number of different types of competitions, the decision was taken to concentrate only on Obedience.

Third, it was thought that the usefulness of the expert system could be enhanced by providing Obedience information about various dog breeds in a database and adding facilities to search for this information. Adding these facilities to the expert system would change the focus from a general Dog Competition Checker to a specific Dog Obedience Advisor.

Therefore, development on the Dog Competition Checker was discontinued. Development continued to create a Dog Obedience Advisor Expert System, which is the subject of discussion throughout the rest of this report.

General Overview of the Dog Obedience Advisor Expert System

The final Dog Obedience Advisor Expert System has two parts: the Dog Obedience Competition Advisor; and the Dog Obedience Breed Advisor.

Dog Obedience Competition Advisor

The Dog Obedience Competition Advisor performs the original function of checking a handler and dog for eligibility and success at Obedience competitions.

It includes two types of Advisors.

- The Pre-Competition Advisor. This makes two checks to determine whether a handler and dog can compete in obedience competitions run by the Royal NSW Canine Council. First, that a handler is a member of the Royal NSW Canine Council. Second, that the handler's dog is at least 6 months old. If either check fails, the handler and dog are ineligible to compete.
- Competition Class Advisors. There is one Competition Class Advisor for each class of obedience competition - Encouragement Class, Novice Class, Open Class, Utility Class, and Obedience Champion. Each Competition Class Advisor checks whether the handler and dog have completed the series of tasks required for the class and is, therefore, eligible for an award or title endorsed by the Australian National Kennel Council.

The task description and tips used by the Advisors in the expert system are summaries and paraphrases from various sources that explain the rules for Obedience competitions [for example, 1, 16, 17].

Dog Obedience Breed Advisor

The Dog Obedience Breed Advisor enables a user to obtain obedience and other information about various dogs breeds from a Sample Dog Breed Database.

The sample database includes information for 63 popular recognised purebreeds compiled from 2002/2003 registration statistics available from the Australian National Kennel Council [2]. It does not include crossbreeds.

The fourteen attributes for each breed in the sample database are loosely divided into five groups:

1 Name Attributes:

- **1** Breed Group. There are seven canine purebreed groups recognised by the Australian National Kennel Council - Gundog, Hound, Nonsporting, Terrier, Toy, Utility, Working.
- **2** Breed Name. Examples are Afghan, Airedale Terrier, Akita, etc.

2 Obedience Attributes

- **3** Obedience Group Rating. This is the ability of the breed's obedience group - Unranked, Low, Fair, Average, High, Excellent, Superior.
- **4** Obedience Group Number. This number corresponds to the breed obedience group rating - 0 (Unranked), 6 (Low), 5 (Fair), 4 (Average), 3 (High), 2 (Excellent), 1 (Superior).
- **5** Obedience Rank. This is the specific rank of the breed. The sample database does not include all ranked breeds. Some breeds have the same rank. Some breeds in the sample database are unranked. In the sample database, values range from 0 (Unranked) through 79.

3 Physical Attributes

- **6** Size. The overall size of the breed - Small, Medium, Large, Extra Large.
- **7** Coat Length. The breed coat length - Short, Medium, Long.
- **8** Grooming Requirements. The amount of grooming required - Low, Medium, High.

4 Personality Attributes

- **9** Activity. The amount of exercise required - Low, Medium, High.
- **10** Temperament. How the breed interacts with others - Placid, Frisky, Aggressive.

11 Nature. The general disposition of the breed - Social, Independent.

5 Tolerance Attributes

12 Tolerance For Young Children. Whether the breed tolerates young children - Yes, No.

13 Tolerance For Animals. Whether the breed tolerates other animals - Yes, No.

14 Tolerance For Strangers. Whether the breed tolerates strangers - Yes, No.

The general information for each breed in the sample database was obtained from a variety of sources [for example, 2, 4, 21].

The obedience information for each breed in the sample database is based on research published by Coren [5] into the intelligence of dogs. Coren distinguishes between three types of intelligence:

- Adaptive Intelligence (learning and problem-solving ability). This is not included in the sample database. This is specific to the individual animal and is measured by canine IQ tests
- Instinctive Intelligence. This is not included in the sample database. This is specific to the individual animal and is measured by canine IQ tests.
- Working/Obedience Intelligence. This is the type of intelligence for which information is included in the sample database. This is breed dependent. Breeds are ranked and placed into obedience groups based on their performance.

Working/obedience intelligence is based on the breed's speed of understanding new commands and responsiveness to obey a new command when first given. This does not necessarily correlate to a specific dog's learning and problem solving ability or its intelligence. For example, a dog may be taught the new command of "Sit". Working/obedience intelligence measures how long it takes dogs of a certain breed to understand the meaning of "Sit" and how quickly the dogs respond to the command "Sit" when first given.

Note The intelligence classification proposed by Coren is not accepted by all dog experts. Nevertheless, the classification provides a useful obedience indicator that was thought to be acceptable for the sample database used by the Dog Obedience Breed Advisor.

The Dog Obedience Breed Advisor provides users with four options to obtain information from the sample database:

- Find a Breed. This option enables the user to type a breed name and display obedience and other information about the breed (if it exists in the sample database).
- Find Breed by Obedience Rank. This option enables the user to type a rank and display obedience and other information about the breed with that rank (if it exists in the sample database).
- Search by All Breed Attributes. This option enables the user to search the sample database for all breeds matching combinations of attributes. The user can search on all attributes (except breed name and obedience rank, which are covered by Find a Breed and Find Breed by Obedience Rank). For every attribute, the user can specify an exact value (small, medium, large, extra large, short, long, low, high, and so on) or specify that any value is acceptable.
- Display the Whole Database. This option enables the user to display the whole sample database. This option is provided primarily so that the user can conveniently find out the contents of the sample database.

Detailed Analysis of the Dog Obedience Advisor Program

This section presents a detailed analysis of the Dog Obedience Advisor program. It describes the user interface and navigation methods, summarises how the program is structured, and systematically walks through the code.

The User Interface and Navigation Methods

The user interface to the Dog Obedience Advisor is text-only, which means that users must type a response to command-line prompts. The interface does not support pointing devices, such as mice and trackballs. There are two types of command-line prompts.

First, prompts using menus. Menus provide the means to navigate within the Dog Obedience Advisor. Using menus, the user can move up and down a hierarchy (for example, to and from a certain Advisor) and exit from the Dog Obedience Advisor. Users select a menu option by typing a number.

For example, from the DOG OBEDIENCE ADVISOR MAIN MENU (shown below), a user has selected option 1 (underlined) for the Dog Obedience Competition Advisor.

```
========================================
   DOG OBEDIENCE ADVISOR MAIN MENU
========================================

1. Dog Obedience Competition Advisor
2. Dog Obedience Breed Advisor
0. Exit

Please select an option (0-2): 1
```

Second, prompts using questions. Questions ask the user to provide information. Depending on the question, an answer might be `y` or `yes` or `n` or `no`, free-form text such as `Beagle` or `BEAGLE`, a number such as `10`, a word such as `gundog` or `any`, or a letter such as `g` or `z`, and so on. Example questions and (underlined) answers include:

```
Are you a member (y/yes | n/no)? y

Breed (for example, Beagle, Pug, Standard Poodle)? Beagle

Rank from 0 (unranked) to 79? 10

Breed Group - there are seven recognised canine purebreed groups in Australia
(g/gundog | h/hound | n/nonsporting | t/terrier | y/toy | u/utility | w/working | z/any)? gundog
```

Program Structure

The programming language used to write the program is the C Language Integrated Production System (CLIPS) v6.20. The program uses approximately 100 rules, four modules, a global variable, and various templates, functions, and facts. Facts are frequently asserted and retracted to control the flow of the program and to store information.

The program is split into four modules - MAIN, COMPETITION, BREED, and EXIT. Within the COMPETITION and BREED modules, the program makes extensive use of control facts to activate rules. This is because a series of rules is often activated to present a series of questions that must be answered before the Dog Obedience Competition Advisor or Dog Obedience Breed Advisor can reach a conclusion.

The MAIN module:
- Initialises the program
- Displays an opening message
- Asks for the name of the user who is using the expert system
- Displays the DOG OBEDIENCE ADVISOR MAIN MENU
- Checks the response to the menu selection
- Either creates a control fact to activate the appropriate rule in another module or, if the response was incorrect, displays an error message and redisplays the DOG OBEDIENCE ADVISOR MAIN MENU

The COMPETITION module:
- Creates templates for four of the five competition classes of Obedience competitions (Encouragement Class, Novice Class, Open Class, and Utility Class - none is needed for Obedience Champion Class)
- Displays the Dog Obedience Competition Advisor Menu
- Checks the response to the menu selection

- Either creates a control fact to activate the appropriate rule or, if the response was incorrect, displays an error message and redisplays the Dog Obedience Competition Advisor Menu
- Asks a series of questions (the questions differ for each competition class)
- For each question, checks the response
- For each question, either creates a fact with the answer or, if the response was incorrect, redisplays the question
- After completing the series of questions, displays a conclusion indicating a successful or unsuccessful final result
- Redisplays the Dog Obedience Competition Advisor Menu

The BREED module:

- Creates a template for a dog breed record, which corresponds to the structure of each record in the Sample Dog Breed Database (which is contained in a separate text file)
- Displays the Dog Obedience Breed Advisor Menu
- Checks the response to the menu selection
- Either creates a control fact to activate the appropriate rule or, if the response was incorrect, displays an error message and redisplays the Dog Obedience Breed Advisor Menu
- Loads the sample database
- Asks one or more questions about a dog breed
- Depending on the user's answers, finds, searches for, and/or displays matching records from the sample database, or displays that no match was found
- Unloads the sample database
- Redisplays the Dog Obedience Breed Advisor Menu

The EXIT module:

- Exits from the expert system

MAIN Module

The MAIN module controls program flow to other modules.

Program Initialisation

The program is initialised and started from the MAIN module. The `init` rule (shown below) is the first rule activated, using the existence of the `initial-fact`. The `init` rule simply activates the `ready` rule.

```
(defrule MAIN::init
 ?f <- (initial-fact)
=>
 (retract ?f)
 (assert (phase ready))
)
```

Opening User Interface

The `ready` rule (shown below) displays the opening user interface. First, it displays an opening message. Second, it asks for the name of the person who is using the Dog Obedience Advisor and stores the name in a global variable (or Anonymous if the user does not enter a name). Finally, it activates the `main-menu` rule.

```
(defrule MAIN::ready
 ?f <- (phase ready)
=>
(retract ?f)
 (printout t crlf
  "DOG OBEDIENCE ADVISOR" crlf
  "=======================================" crlf
  "This expert system is a Dog Obedience Advisor, offering obedience" crlf
  "advice mainly for residents of New South Wales, Australia. It has:" crlf
  "two functions:" crlf
...
  "Before we start, please type your name (press RETURN for Anonymous): ")
 (bind ?*handler* (readline))
 (if (eq (str-length ?*handler*) 0)
  then (bind ?*handler* "Anonymous")
 );if
 (printout t crlf "Welcome " ?*handler* " to the DOG OBEDIENCE ADVISOR." crlf)
 (assert (phase main-menu))
)
```

DOG OBEDIENCE ADVISOR MAIN MENU

The `main-menu` rule (see the code segment below) displays the DOG OBEDIENCE ADVISOR MAIN MENU, which asks the user to select an option:

```
...
 (printout t crlf
  "=======================================" crlf
  "   DOG OBEDIENCE ADVISOR MAIN MENU" crlf
  "=======================================" crlf
  crlf
  "1. Dog Obedience Competition Advisor" crlf
  "2. Dog Obedience Breed Advisor" crlf
  "0. Exit" crlf
  crlf
  "Please select an option (" ?min "-" ?max "): "
 );printout
 (bind ?option (readline)) (bind ?option (readline))
 (if (check-menu ?min ?max ?option)
  then (and (printout t crlf "Invalid option " ?option crlf)
            (assert (phase main-menu)))
  else (and (bind ?option (convert-main-menu (string-to-field ?option)))
            (assert (phase ?option))))
)
...
```

Error Checking and Switching

The `check-menu` function (shown below) used by the `main-menu` rule performs three checks to validate the main menu selection. First, that a value exists. Second, that the value is an integer. Finally, that the value is within the minimum and maximum range of valid integers (for the MAIN MENU this is from 0 through 2).

```
(deffunction MAIN::check-menu (?min ?max ?option)
 (if (or (eq (str-length ?option) 0)
         (not (integerp (string-to-field ?option)))
         (< (string-to-field ?option) ?min)
         (> (string-to-field ?option) ?max))
     then TRUE
     else FALSE))
```

If the main menu selection is invalid, an error message is displayed, which shows the error, and the main menu is redisplayed.

If the main menu selection is valid, the `convert-main-menu` function (shown below) uses a `switch` construct to convert the integer to a string that determines the module - COMPETITION, BREED, or EXIT - to which control is passed.

```
(deffunction MAIN::convert-main-menu (?option)
 (switch ?option
  (case 1 then (bind ?return-value competition))
  (case 2 then (bind ?return-value breed))
  (case 0 then (bind ?return-value exit)))
 (return ?return-value))
```

Note The `check-menu` function and similar `convert` functions are used by other modules.

COMPETITION Module

The COMPETITION module handles all enquiries about obedience competitions.

Templates for Competition Classes

The COMPETITION module first creates templates for the various competition classes. The following template is used for Encouragement Class (templates for other competition classes are similar). Each slot in a template represents a required competition task. All templates use the `allowed-values` and `default` constructs to control acceptable values.

```
(deftemplate COMPETITION::encouragement-class
 (slot heel-on-lead (allowed-values Passed Failed) (default Passed))
 (slot stand-for-examination (allowed-values Passed Failed) (default Passed))
 (slot stand-stay (allowed-values Passed Failed) (default Passed))
 (slot sit-stay (allowed-values Passed Failed) (default Passed))
 (slot down-stay (allowed-values Passed Failed) (default Passed))
 (slot total-points (allowed-values Passed Failed) (default Passed))
 (slot result (allowed-values Passed Failed) (default Passed))
)
```

Dog Obedience Competition Advisor Menu

The Dog Obedience Competition Advisor Menu (see the code segment below) enables the user to select the Pre-Competition Advisor or a Competition Class Advisor:

```
...
 (printout t crlf
  "----------------------------------------" crlf
  "1. Dog Obedience Competition Advisor Menu" crlf
  "----------------------------------------" crlf
  "The Dog Obedience Competition Advisor enables you to obtain" crlf
  "advice about obedience competitions and titles endorsed by the" crlf
  "Royal New South Wales Canine Council, Australia." crlf
  crlf
  "1. Pre-Competition Advisor" crlf
  "2. Encouragement Class Advisor" crlf
  "3. Novice Class (CD) Advisor" crlf
  "4. Open Class (CDX) Advisor" crlf
  "5. Utility Class (UD) Advisor" crlf
  "6. Obedience Champion (OC) Advisor" crlf
  "0. Return to the MAIN MENU" crlf
  crlf
  "Please select an option (" ?min "-" ?max "): ")
 (bind ?option (readline))
...
```

General Advisor Features

Each Advisor works in the same way. It asks a series of questions to which the user must answer either `yes` or `no`. Two functions (see below) taken from Giarratano and Riley [10] check the answer. If the answer is incorrect or invalid, the question is repeated.

```
(deffunction MAIN::ask-question (?question $?allowed-values)
   (printout t ?question)
   (bind ?answer (read))
   (if (lexemep ?answer)
       then (bind ?answer (lowcase ?answer)))
   (while (not (member ?answer ?allowed-values)) do
      (printout t ?question)
      (bind ?answer (read))
      (if (lexemep ?answer)
          then (bind ?answer (lowcase ?answer))))
   ?answer)
(deffunction MAIN::yes-or-no-p (?question)
   (bind ?response (ask-question ?question yes no y n))
   (if (or (eq ?response yes) (eq ?response y))
       then TRUE
       else FALSE))
```

Pre-Competition Advisor

The Pre-Competition Advisor makes two checks. First, that a handler is a member of the Royal NSW Canine Council. Second, that the handler's dog is at least 6 months old. As the following code segments show, if the answer to either question is `no`, the Advisor concludes that it is not possible for the handler and dog to compete.

```
...
 (printout t crlf
  "--- Check Handler's Membership ---" crlf
  "The handler must be a member of the Royal NSW Canine Council to" crlf
  "participate in obedience competitions run the Council and for" crlf
  "their dog to be eligible for awards and titles from the Council." crlf
  crlf
 );printout
 (if (yes-or-no-p "Are you a member (y/yes | n/no)? ")
  then (printout t crlf
        "    --- Result ---" crlf
        "    Congratulations " ?*handler* ". You are eligible to compete." crlf
       );printout
  else (printout t crlf
        "    --- Result ---" crlf
        "    Sorry " ?*handler* ". You must join the Royal NSW Canine Council before" crlf
        "    you can compete in any obedience competition run by the Council." crlf
        "    You can download a membership form from http://www.rnswcc.org.au/." crlf
       );printout
 )
...
 (printout t crlf
  "--- Check Dog's Minimum Age ---" crlf
  "The dog must be at least 6 months old to compete in obedience competitions." crlf
  crlf
 );printout
 (if (yes-or-no-p "Is the dog at least 6 months old (y/yes | n/no)? ")
  then (printout t crlf
        "    --- Result ---" crlf
        "    Congratulations " ?*handler* ". The dog is old enough to compete." crlf
       );printout
  else (printout t crlf
        "    --- Result ---" crlf
        "    The dog is too young to compete in obedience competitions." crlf
        "    A dog less than 6 months is unlikely to have the physical or mental" crlf
        "    ability to complete the required exercises. Also, the dog must be" crlf
        "    fully vaccinated before competing, and final vaccinations cannot be" crlf
        "    administered until the dog is at least 6 months old." crlf
       );printout
 );if
...
```

Competition Class Advisors

All Competition Class Advisors first check whether the dog already has an award or title in that class. If so, the dog is not eligible to compete in that class again. The following code segment is for Encouragement Class:

```
...
 (printout t crlf
  "--- Check Awards and Titles ---" crlf
  "The dog cannot compete in Encouragement Class if it already" crlf
  "has an Encouragement Class award or a Novice, Open, Utility," crlf
  "or Champion title." crlf
  crlf
 );printout
 (if (yes-or-no-p "Does the dog already have one of these awards or titles (y/yes | n/no)? ")
  then (and (printout t crlf
             "   --- Result ---" crlf
             "   The dog cannot compete in Encouragement Class." crlf
             "   The dog can and should enter the appropriate competition." crlf
            );printout
            (assert (phase competition-finished))
       );and
  else (assert (phase competition-encouragement-begin-to-check-tasks))
 );if
...
```

All Competition Class Advisors then ask a series of questions to which the user must answer either `yes` or `no`. Each question checks whether the handler and dog has passed a required task for that class.

One such Encouragement Class task, for example, is Heel on Lead (see the code segment below). The Competition Class Advisor briefly describes the task and asks the handler whether the dog has passed. If the answer is `yes`, the Competition Class Advisor records that the dog has passed the task. If the answer is `no`, it records that the dog has failed and the Advisor displays some tips.

```
...
 (printout t crlf
  "--- Heel on Lead ---" crlf
  "The dog must walk briskly next to the handler on a loose lead in different" crlf
  "directions and at various speeds. Includes down, stand, and figure eight." crlf
  "When the judge commands Halt, the dog sits automatically." crlf
  crlf
 );printout
 (if (yes-or-no-p "Can the dog Heel on Lead (15/30 points) (y/yes | n/no)? ")
  then ;default is (heel-on-lead Passed)
  else (and (printout t crlf
             "    --- Tips ---" crlf
             "    The dog must heel steadily next to the handler's left foot." crlf
             "    The dog must promptly obey each command." crlf
             "    The dog must not anticipate any command." crlf
             "    The handler must not adapt his/her speed to the dog's speed." crlf
             "    The handler must not continually tug the dog with the lead." crlf
             "    The handler must hold the lead in the correct hand." crlf
            );printout
            (modify ?t (heel-on-lead Failed) (result Failed))
       );and
 );if
...
```

The default for every task fact and `result` fact is `Passed` (this is defined in the template for each class). Therefore, passing a task requires no change to the information about the task. For example, the default for the `heel-on-lead` task fact is `Passed`. Therefore, passing the Heel on Lead task requires no change to the `heel-on-lead` task fact.

Failing is task is recorded in two ways. First, the fact for the task is modified to register the failure, for example, the `heel-on-lead` task fact is modified to `Failed`. Second, the `result` fact is modified to `Failed`.

The `result` fact is a flag. It is used by the Competition Class Advisor after it has asked all questions to determine the final result. This is because of the basic rule used to determine the final result for each class:

> If the handler and dog pass all tasks, then the handler and dog pass the competition class.

In other words:

> If the handler and dog fail any task, then the handler and dog fail the competition class.

After completing the questions, the Competition Class Advisor displays a summary of the results for each task in the class and the final result (see the code segment below):

```
...
 (printout t crlf
  "--- Summary ---" crlf
  "You have provided the following Encouragement Class information:" crlf
  "Heel on Lead ........ " ?heel-on-lead crlf
  "Stand for Examination " ?stand-for-examination crlf
  "Stand Stay .......... " ?stand-stay crlf
  "Sit Stay ............ " ?sit-stay crlf
  "Down Stay ........... " ?down-stay crlf
  "Total Points ........ " ?total-points crlf
  crlf
  "--- FINAL RESULT ---  " ?result crlf
 );printout
...
```

The program uses the value of the `result` fact to display a final message for the user of the Competition Class Advisor as the following code segment for Encouragement Class shows:

```
...
 (if (eq ?result Passed)
  then (assert (phase competition-encouragement-succeeded))
  else (assert (phase competition-encouragement-failed))
 );if
```

```
...
```

For example, the final `Passed` message for Encouragement Class is:

```
...
 (printout t crlf
  "--- Conclusion ---" crlf
  "Congratulations " ?*handler* ". The dog is eligible for an Encouragement Class award." crlf
  "Next level is Novice Class." crlf
 );printout
...
```

The final `Failed` message for Encouragement Class is:

```
...
 (printout t crlf
  "--- Conclusion ---" crlf
  "Sorry " ?*handler* ". The dog is not eligible for an Encouragement Class award." crlf
  "The dog must pass every task AND receive a total score of 115/130 points in a" crlf
  "single competition. Practise the tasks that the dog failed and please keep trying." crlf
 );printout
...
```

Redisplaying the Dog Obedience Competition Advisor Main Menu

Finally, the user is asked to press the RETURN key to redisplay the Dog Obedience Competition Advisor Main Menu (see the code segment below) from which the user can then select another Advisor or return to the MAIN MENU:

```
(defrule COMPETITION::phase-competition-finished
 ?f <- (phase competition-finished)
=>
 (retract ?f)
 (printout t crlf "Press the RETURN key to continue ")
 (bind ?option (readline))
 (assert (phase competition))
)
```

BREED Module

The BREED module handles all obedience and other enquiries about dog breeds in the sample database.

Template for a Dog Breed Record

The BREED module first creates a template for a dog breed record (see below):

```
...
(deftemplate BREED::dog
 (slot group) (multislot breed)
 (slot orat) (slot ogrp) (slot orank)
 (slot size) (slot coat) (slot groom)
 (slot activity) (slot temperament) (slot nature)
 (slot kidok) (slot aniok) (slot strok)
)
...
```

The meaning of each slot or multislot in the `dog` template is:

`(slot group)`
Specifies the name of the group in which the breed falls. Sample database values are `gundog`, `hound`, `nonsporting`, `terrier`, `toy`, `utility`, `working`. At the appropriate prompt, a user can specify `g`, `gundog`, `h`, `hound`, `n`, `nonsporting`, `t`, `terrier`, `y`, `toy`, `u`, `utility`, `w`, `working`, `z`, `any`.

`(multislot breed)`
Specifies the name of the dog breed. Sample database examples are `afghan`, `airedale terrier`, `akita`, etc.

Note The `multislot` construct is used for `breed` because the name of the dog breed may have more than one word, for example, `airedale terrier`. Later in the program, the `explode$` and `implode$` constructs are used when obtaining and displaying dog breed names.

`(slot orat)`
Specifies the breed obedience group rating. Sample database values are `unranked`, `low`, `fair`, `average`, `high`, `excellent`, `superior`. At the appropriate prompt, a user can specify `u`, `unranked`, `l`, `low`, `f`, `fair`, `a`, `average`, `h`, `high`, `e`, `excellent`, `s`, `superior`, `z`, `any`.

`(slot ogrp)`
Specifies the breed obedience group number. Sample database values are `0` (unranked), `6`, `5`, `4`, `3`, `2`, `1`. Users do not specify a value (because the number corresponds to the breed obedience group rating that users can already specify).

`(slot orank)`
Specifies the breed obedience rank. Sample database values range from `0` (unranked) through `79`. At the appropriate prompt, a user can specify an integer from `0` through `79`.

`(slot size)`
Specifies the breed size. Sample database values are `s`, `m`, `l`, `x`. Users can specify `s`, `small`, `m`, `medium`, `l`, `large`, `x`, `xlarge`, `z`, `any`.

`(slot coat)`
Specifies the breed coat length. Sample database values are `s`, `m`, `l`. At the appropriate prompt, a user can specify `s`, `short`, `m`, `medium`, `l`, `long`, `z`, `any`.

`(slot groom)`
Specifies the amount of grooming required. Sample database values are `l`, `m`, `h`. At the appropriate prompt, a user can specify `l`, `low`, `m`, `medium`, `h`, `high`, `z`, `any`.

`(slot activity)`
Specifies the amount of exercise required. Sample database values are `l`, `m`, `h`. At the appropriate prompt, a user can specify `l`, `low`, `m`, `medium`, `h`, `high`, `z`, `any`.

`(slot temperament)`
Specifies how the breed interacts with others. Sample database values are `placid`, `frisky`, `aggressive`. At the appropriate prompt, a user can specify `p`, `placid`, `f`, `frisky`, `a`, `aggressive`, `z`, `any`.

`(slot nature)`
Specifies the general disposition of the breed. Sample database values are `soc`, `ind`. At the appropriate prompt, a user can specify `s`, `soc`, `social`, `i`, `ind`, `independent`, `z`, `any`.

`(slot kidok)`
: Specifies the breed's tolerance for young children. Sample database values are `y`, `n`. At the appropriate prompt, a user can specify `y`, `yes`, `n`, `no`, `z`, `any`.

`(slot aniok)`
: Specifies the breed's tolerance for other animals. Sample database values are `y`, `n`. At the appropriate prompt, a user can specify `y`, `yes`, `n`, `no`, `z`, `any`.

`(slot strok)`
: Specifies the breed's tolerance for strangers. Sample database values are `y`, `n`. At the appropriate prompt, a user can specify `y`, `yes`, `n`, `no`, `z`, `any`.

Dog Obedience Breed Advisor Menu

The Dog Obedience Breed Advisor Menu (see the code segment below) enables the user to select the method used to find, search, or display records in the sample database:

```
...
 (printout t crlf
  "----------------------------------------" crlf
  "2. Dog Obedience Breed Advisor Menu" crlf
  "----------------------------------------" crlf
  "The Dog Obedience Breed Advisor enables you to obtain" crlf
  "obedience and other information about various dogs breeds" crlf
  "using a Sample Dog Breed Database." crlf
  crlf
  "1. Find a Breed" crlf
  "2. Find Breed by Obedience Rank" crlf
  "3. Search by All Breed Attributes" crlf
  "4. Display the Whole Database" crlf
  "0. Return to the MAIN MENU" crlf
  crlf
  "Please select an option (" ?min "-" ?max "): "
 );printout
...
```

Loading the Sample Database

When the user selects a menu option, the program (after error-checking) attempts to load the sample database (see the code segment below). If the sample database does not exist in the same folder as the program, the program displays an error message before asking the user to press the RETURN key to redisplay the Dog Obedience Breed Advisor Main Menu.

```
...
 (if (check-menu ?min ?max ?option)
 then (and (printout t crlf "Invalid option " ?option crlf)
           (assert (phase breed))
      );and
 else (if (eq (string-to-field ?option) 0)
       then (assert (phase breed-return-to-main-menu))
       else (if (load-facts "dogoaes.db") ;dog database currently contains 63 breeds
             then
                  (and
                       (bind ?option (convert-breed-menu (string-to-field ?option)))
                       (assert (phase (sym-cat breed- ?option)))
                  );and
             else
                  (and
                      (printout t crlf
                       "*** SAMPLE DOG BREED DATABASE NOT FOUND ***" crlf
                       " Please restore the database and try again" crlf
                      );printout
                      (retract *)
                      (assert (phase breed-finished))
                 );and
            );if
      );if
 );if
...
```

Find a Breed Option

The Find a Breed option (see the code segment below) enables the user to specify and find a dog breed. The user can use uppercase or lowercase letters. The entry must match exactly (partial matches are not supported).

```
...
 (printout t crlf
  "----------------------------------------" crlf
  "2.1. Find a Breed" crlf
  "----------------------------------------" crlf
  "Find a breed and display its obedience and other characteristics." crlf
  "You can use uppercase or lowercase letters." crlf
  "Your entry must match exactly (partial matches are not supported)." crlf
  "You can also Display the Whole Database to see all breeds." crlf
  crlf
  "Breed (for example, Beagle, Pug, Standard Poodle)? "
 );printout
 (bind ?breed (readline))
 (bind ?breed (lowcase ?breed))
 (bind ?breed (explode$ ?breed))
...
```

Find a Breed Option - Match Found

If a match is found, the breed's obedience and other characteristics are displayed in a table (see the code segment below):

```
...
 (breed $?breed)
 (dog (group ?group) (breed $?breed)
      (orat ?orat) (ogrp ?ogrp) (orank ?orank)
      (size ?size) (coat ?coat) (groom ?groom)
      (activity ?activity) (temperament ?temperament) (nature ?nature)
      (kidok ?kidok) (aniok ?aniok) (strok ?strok)
 );dog
=>
 (printout t
  "----------------------------------------"
  "----------------------------------------"
  "----------------------------------------"
  "----------------------------------------"
  "----------------------------------------"
  crlf
 );printout
 (format t "%-11s | %-30s | %-9s | %-5d | %-4d | %-4s | %-4s | %-5s | %-8s | %-11s | %-6s | %-5s | %-6s | %-8s |
%-30s | %-11s%n"
          ?group (implode$ ?breed) ?orat ?ogrp ?orank ?size ?coat ?groom ?activity ?temperament ?nature ?kidok
?aniok ?strok (implode$ ?breed) ?group
 );format
...
```

Find Breed by Obedience Rank Option

The Find Breed by Obedience Rank option (see the code segment below) enables the user to specify an obedience rank and find the dog breed(s) with the specified rank. The value must be a number.

```
...
 (printout t crlf
  "----------------------------------------" crlf
  "2.2. Find Breed by Obedience Rank" crlf
  "----------------------------------------" crlf
  "Find a rank and display the breed with that obedience rank." crlf
  "The sample database does not include all ranked breeds." crlf
  "Some breeds are unranked. Some breeds share the same rank." crlf
  "You can also Display the Whole Database to see all breeds." crlf
  crlf
  "Ranks are based on research published by Stanley Coren in" crlf
  "'The Intelligence of Dogs', Bantam Books, 1995, ISBN 0553374524." crlf
  crlf
  "This is described as breed-dependent working/obedience" crlf
  "intelligence based on the breed's speed of understanding new" crlf
  "commands and responsiveness to obey a new command when first" crlf
  "given. This does not necessarily correlate to a specific dog's" crlf
  "learning and problem solving ability or its intelligence." crlf
  crlf
  "Rank from 0 (unranked) to 79? "
 );printout
 (bind ?orank (read))
 (while (or
            (not (integerp ?orank))
            (< ?orank 0)
            (> ?orank 79)
        );or
        do
        (printout t
         "Rank from 0 (unranked) to 79? "
        );printout
        (bind ?orank (read))
        ;do
 );while
...
```

Find Breed by Obedience Rank Option - Match Found

If a breed or breeds with the specified rank is found, the breed's obedience and other characteristics are displayed in a table (see the code segment below):

```
...
 (orank ?orank)
 (dog (group ?group) (breed $?breed)
      (orat ?orat) (ogrp ?ogrp) (orank ?orank)
      (size ?size) (coat ?coat) (groom ?groom)
      (activity ?activity) (temperament ?temperament) (nature ?nature)
      (kidok ?kidok) (aniok ?aniok) (strok ?strok)
 );dog
=>
 (printout t
  "-----------------------------------------"
  "-----------------------------------------"
  "-----------------------------------------"
  "-----------------------------------------"
  "-----------------------------------------"
  crlf
 );printout
 (format t "%-11s | %-30s | %-9s | %-5d | %-4d | %-4s | %-4s | %-5s | %-8s | %-11s | %-6s | %-5s | %-6s | %-8s |
%-30s | %-11s%n"
           ?group (implode$ ?breed) ?orat ?ogrp ?orank ?size ?coat ?groom ?activity ?temperament ?nature ?kidok
?aniok ?strok (implode$ ?breed) ?group
 );format
...
```

Display the Whole Database Option

The Display the Whole Database option (see the code segment below) enables the user to display all records in the sample database. This option is provided primarily so that a user can conveniently find out the contents of the sample database.

```
...
 (printout t crlf
  "-----------------------------------------" crlf
  "2.4. Display the Whole Database" crlf
  "-----------------------------------------" crlf
  "Display all records in the database." crlf
  "This option is provided primarily so that you can" crlf
  "conveniently find out the contents of the database." crlf
  crlf
  "--- WARNING ---" crlf
  "The database contains a lot of records." crlf
  crlf
 );printout
 (if (yes-or-no-p "Do you want to continue (y/yes | n/no)? ")
  then (and (assert (display-all))
            (assert (phase breed-display-the-results))
       );and
  else (assert (phase breed-finished))
 );if
```

```
...
```

Find and Display Options - No Match Found

If the Find a Breed, Find Breed by Obedience Rank, or Display the Whole Database option does not find any matching record(s), the following code segment displays `*** NO MATCH FOUND ***` in a table:

```
...
 (not
      (and
           (or
               (breed $?breed)
               (orank ?orank)
               (search-by-all)
               (display-all)
           );or
           (dog (group ?group) (breed $?breed)
                (orat ?orat) (ogrp ?ogrp) (orank ?orank)
                (size ?size) (coat ?coat) (groom ?groom)
                (activity ?activity) (temperament ?temperament) (nature ?nature)
                (kidok ?kidok) (aniok ?aniok) (strok ?strok)
           );dog
      );and
 );not
=>
 (printout t
  "----------------------------------------"
  "----------------------------------------"
  "----------------------------------------"
  "----------------------------------------"
  "----------------------------------------"
  crlf
 );printout
 (printout t crlf
  "*** NO MATCH FOUND ***" crlf
  crlf
 );printout
...
```

Search by All Breed Attributes Option

The Search by All Breed Attributes option enables the user to search the sample database for dog breeds that match a combination of attributes. A series of questions is asked in separate rules to which the user can specify an exact value or `z` or `any` to specify a wildcard match. The following questions are asked in separate rules:

```
...
 (printout t crlf
  "Breed Group - there are 7 recognised purebreed groups" crlf
  "(g/gundog | h/hound | n/nonsporting | t/terrier | y/toy | u/utility | w/working | z/any)? "
 );printout
...
 (printout t crlf
  "Obedience Group Rating" crlf
  "(s/superior | e/excellent | h/high | a/average | f/fair | l/low | u/unranked | z/any)? "
 );printout
...
 (printout t crlf
  "Breed Size - affects food costs, living area, dog's life span" crlf
  "(s/small | m/medium | l/large | x/xlarge | z/any)? "
 );printout
...
 (printout t crlf
  "Coat Length - affects shedding, grooming, allergies, cleaning" crlf
  "(s/short | m/medium | l/long | z/any)? "
 );printout...
...
 (printout t crlf
  "Grooming Requirements - affects time required for upkeep, cleaning" crlf
  "(l/low | m/medium | h/high | z/any)? "
 );printout...
...
 (printout t crlf
  "Activity Level - affects exercise requirements, yard size, your lifestyle" crlf
  "(l/low | m/medium | h/high | z/any)? "
 );printout...
...
 (printout t crlf
  "Temperament - affects how the dogs interacts with other dogs and humans" crlf
  "(p/placid | f/frisky | a/aggressive | z/any)? "
 );printout
...
 (printout t crlf
  "Nature - affects the extent to which the dog needs companionship" crlf
  "(s/social | i/independent | z/any)? "
 );printout
...
 (printout t crlf
  "Child Tolerance - is it important for the breed to tolerate children?" crlf
  "(y/yes | n/no | z/any)? "
 );printout
...
 (printout t crlf
  "Animal Tolerance - is it important for the breed to tolerate other animals?" crlf
  "(y/yes | n/no | z/any)? "
 );printout
...
 (printout t crlf
  "Stranger Tolerance - is it important for the breed to tolerate strangers?" crlf
  "(y/yes | n/no | z/any)? "
 );printout
...
```

The rule for each question includes error checking (separate rules are not used to check errors to minimise the number of rules in the program). For example, the Breed Group rule (see the code segment below) uses an `if ... then` construct to check whether the user has typed a number and, if not, converts alphabetic characters to lowercase. A `while` construct then checks that the user has typed a valid value. If not, the question is repeated. If so, a `switch` construct converts the user's value to the correct form of the value used in the sample database.

```
...
 (bind ?group (read))
 (if (not (numberp ?group)) then (bind ?group (lowcase ?group)))
 (while
        (not
             (or
                 (eq ?group g) (eq ?group gundog)
                 (eq ?group h) (eq ?group hound)
                 (eq ?group n) (eq ?group nonsporting)
                 (eq ?group t) (eq ?group terrier)
                 (eq ?group y) (eq ?group toy)
                 (eq ?group u) (eq ?group utility)
                 (eq ?group w) (eq ?group working)
                 (eq ?group z) (eq ?group any)
             );or
        );not
        do
           (printout t
            "(g/gundog | h/hound | n/nonsporting | t/terrier | y/toy | u/utility | w/working | z/any)? "
           );printout
           (bind ?group (read))
          (if (not (numberp ?group)) then (bind ?group (lowcase ?group)))
        ;do
 );while
 (switch ?group
  (case g then (bind ?group gundog))
  (case h then (bind ?group hound))
  (case n then (bind ?group nonsporting))
  (case t then (bind ?group terrier))
  (case y then (bind ?group toy))
  (case u then (bind ?group utility))
  (case w then (bind ?group working))
  (case any then (bind ?group z))
 );switch

...
```

Search by All Breed Attributes Option - Match Found

If the Search by All Breed Attributes option finds one or more matching record(s), the matching breed's obedience and other characteristics are displayed in a table (see the code segment below):

```
...
 (group ?temp-group)
 (orat ?temp-orat)
 (size ?temp-size)
 (coat ?temp-coat)
 (groom ?temp-groom)
 (activity ?temp-activity)
 (temperament ?temp-temperament)
 (nature ?temp-nature)
 (kidok ?temp-kidok)
 (aniok ?temp-aniok)
 (strok ?temp-strok)
 (dog (group ?group&:(or (eq ?group ?temp-group) (eq ?temp-group z)))
      (breed $?breed)
      (orat ?orat&:(or (eq ?orat ?temp-orat) (eq ?temp-orat z)))
      (ogrp ?ogrp)
      (orank ?orank)
      (size ?size&:(or (eq ?size ?temp-size) (eq ?temp-size z)))
      (coat ?coat&:(or (eq ?coat ?temp-coat) (eq ?temp-coat z)))
      (groom ?groom&:(or (eq ?groom ?temp-groom) (eq ?temp-groom z)))
      (activity ?activity&:(or (eq ?activity ?temp-activity) (eq ?temp-activity z)))
      (temperament ?temperament&:(or (eq ?temperament ?temp-temperament) (eq ?temp-temperament z)))
      (nature ?nature&:(or (eq ?nature ?temp-nature) (eq ?temp-nature z)))
      (kidok ?kidok&:(or (eq ?kidok ?temp-kidok) (eq ?temp-kidok z)))
      (aniok ?aniok&:(or (eq ?aniok ?temp-aniok) (eq ?temp-aniok z)))
      (strok ?strok&:(or (eq ?strok ?temp-strok) (eq ?temp-strok z)))
 );dog
=>
 (printout t
  "-----------------------------------------"
  "-----------------------------------------"
  "-----------------------------------------"
  "-----------------------------------------"
  "-----------------------------------------"
  crlf
 );printout
 (format t "%-11s | %-30s | %-9s | %-5d | %-4d | %-4s | %-4s | %-5s | %-8s | %-11s | %-6s | %-5s | %-6s | %-8s |
%-30s | %-11s%n"
           ?group (implode$ ?breed) ?orat ?ogrp ?orank ?size ?coat ?groom ?activity ?temperament ?nature ?kidok
?aniok ?strok (implode$ ?breed) ?group
 );format
...
```

Search by All Breed Attributes - No Match Found

If Search by All Breed Attributes does not find any matching record(s), the following code segment displays `*** NO MATCH FOUND ***` in a table:

```
...
 (not
      (and
           (group ?temp-group)
           (orat ?temp-orat)
           (size ?temp-size)
           (coat ?temp-coat)
           (groom ?temp-groom)
           (activity ?temp-activity)
           (temperament ?temp-temperament)
           (nature ?temp-nature)
           (kidok ?temp-kidok)
           (aniok ?temp-aniok)
           (strok ?temp-strok)
           (dog (group ?group&:(or (eq ?group ?temp-group) (eq ?temp-group z)))
                (breed $?breed)
                (orat ?orat&:(or (eq ?orat ?temp-orat) (eq ?temp-orat z)))
                (ogrp ?ogrp)
                (orank ?orank)
                (size ?size&:(or (eq ?size ?temp-size) (eq ?temp-size z)))
                (coat ?coat&:(or (eq ?coat ?temp-coat) (eq ?temp-coat z)))
                (groom ?groom&:(or (eq ?groom ?temp-groom) (eq ?temp-groom z)))
                (activity ?activity&:(or (eq ?activity ?temp-activity) (eq ?temp-activity z)))
                (temperament ?temperament&:(or (eq ?temperament ?temp-temperament) (eq ?temp-temperament z)))
                (nature ?nature&:(or (eq ?nature ?temp-nature) (eq ?temp-nature z)))
                (kidok ?kidok&:(or (eq ?kidok ?temp-kidok) (eq ?temp-kidok z)))
                (aniok ?aniok&:(or (eq ?aniok ?temp-aniok) (eq ?temp-aniok z)))
                (strok ?strok&:(or (eq ?strok ?temp-strok) (eq ?temp-strok z)))
           );dog
      );and
 );not
=>
 (printout t
  "----------------------------------------"
  "----------------------------------------"
  "----------------------------------------"
  "----------------------------------------"
  "----------------------------------------"
  crlf
 );printout
 (printout t crlf
  "*** NO MATCH FOUND ***" crlf
  crlf
 );printout
...
```

Displaying the Results From a Sample Database Query

The results from every sample database query are displayed in a table by a group of rules. The first rule that is activated in the group displays the heading to the table. Then one of the rules (discussed above) that displays matching records or *** NO MATCH FOUND *** is activated. Finally, the last rule in the group displays the footer to the table.

Salience plus judicious pattern matching is used to control the correct order of activation in the group of rules. The first rule in the group uses the highest salience of 100. Rules that display matching records or *** NO MATCH FOUND *** use salience of 90. The last rule in the group uses the lowest salience of 80.

The following code segment shows the first rule in the group (salience 100). Two `printout` constructs displays separator lines. Three `format` constructs displays the text of the table heading.

```
...
 (declare (salience 100))
 (phase breed-display-the-results)
=>
 (printout t crlf
  "----------------------------------------"
  "----------------------------------------"
  "----------------------------------------"
  "----------------------------------------"
  "----------------------------------------"
  crlf
 );printout
 (format t "--- SAMPLE DOG BREED DATABASE --- %30s %23s %30s %27s%n"
                            Obedience         Features            Attributes               Tolerance
 );format
 (format t "%71s %21s %33s %27s%n"
            ------------------------ ------------------- ------------------------------- -----------------------
--
 );format
 (format t "%-11s | %-30s | %-9s | %-5s | %-4s | %-4s | %-4s | %-5s | %-8s | %-11s | %-6s | %-5s | %-6s | %-8s |
%-30s | %-11s%n"
;                       --- Obedience --- --- Features -- -------- Attributes ------- ----- Tolerance -----
            Group Breed Rating Group Rank Size Coat Groom Activity Temperament Nature Child Animal Stranger
Breed Group
 );format
...
```

The following code segment shows the last rule in the group (salience 80). This rule displays the separator line and, because the width of the table exceeds the width of the screen, a message to use the mouse to scroll to the RHS of the screen to see the rest of the table.

```
...
 (declare (salience 80))
 (phase breed-display-the-results)
=>
```

```
 (printout t
  "-----------------------------------------"
  "-----------------------------------------"
  "-----------------------------------------"
  "-----------------------------------------"
  "-----------------------------------------"
  crlf
 );printout
 (printout t
  ">>>>>>>>>>>>>>>> TABLE CONTINUES OFFSCREEN "
  ">>>>>>>>>>>>>>>> SCROLL USING YOUR MOUSE "
  ">>>>>>>>>>>>>>>>"
  crlf
 );printout
(assert (phase breed-finished))
...
```

Unloading the Sample Database

After finding, searching, or displaying, the program retracts all current facts (see the code segment below). This unloads the sample database:

```
(defrule BREED::phase-breed-finished
 ?f <- (phase breed-finished)
=>
 (retract *)
 ...
)
```

Redisplaying the Dog Obedience Breed Advisor Main Menu

Finally, the user is asked to press any key to redisplay the Dog Obedience Breed Advisor Main Menu (see the code segment below):

```
(defrule BREED::phase-breed-finished
 ?f <- (phase breed-finished)
=>
 ...
 (printout t crlf "Press any key to continue ")
 (bind ?option (readline))
 (assert (phase breed))
)
```

EXIT Module

The EXIT module has one rule that displays a friendly closing message and exits from the expert system (see below):

```
(defrule EXIT::phase-exit-dog-obedience-advisor
 (declare (auto-focus TRUE))
 ?f <- (phase exit-dog-obedience-advisor)
=>
 (retract ?f)
 (printout t crlf
  "Thank you " ?*handler* " for using the DOG OBEDIENCE ADVISOR." crlf
  crlf
 );printout
 (assert (stop))
 (assert (quit))
)
```

Problems Encountered While Building the Dog Obedience Advisor

This section discusses problems encountered while building the Dog Obedience Advisor.

Dog Obedience Competition Advisor

Some problems were encountered while building the Dog Obedience Competition Advisor.

General or Specific Obedience Class Templates

One problem was whether to use one general template, that incorporated all tasks for all obedience classes, or a number of specific templates, that described the tasks required at each specific class of Obedience competition.

To understand the similarities and differences among tasks at each class, consider the following table:

	Encouragement	**Novice**	**Open**	**Utility**	**Champion**
Heel on Lead	Pass: 15/30 On lead	Pass: 15/30 On lead	Not applicable	Not applicable	Not applicable
Heel Free	Not applicable	Pass: 20/40 Off lead	Pass: 15/30 Off lead	Not applicable	Not applicable
Stand for Examination	Pass: 10/20 On lead	Pass: 10/20 On lead	Pass: 10/20 Off lead	Pass: 5/10	Pass: 5/10
Stand Stay	Pass: 10/20 On lead	Pass: 10/20 Off lead 5 metres	Not applicable	Not applicable	Not applicable
Recall	Not applicable	Pass: 15/30 Off lead 12 metres	Not applicable	Not applicable	Not applicable
Sit Stay	Pass: 15/30 On lead	Pass: 15/30 Off lead 1 minute 12 metres	Pass: 13/25 Off lead 3 minutes No handler	Not applicable	Not applicable
Down Stay	Pass: 15/30 On lead	Pass: 15/30 Off lead 3 minutes 12 metres	Pass: 13/25 Off lead 5 minutes No handler	Pass: 13/25 Off lead 10 minutes No handler	Pass: 13/25 Off lead 10 minutes No handler
Drop on Recall	Not applicable	Not applicable	Pass: 15/30 Off lead	Not applicable	Not applicable
Retrieve Dumbbell on Flat	Not applicable	Not applicable	Pass: 10/20	Not applicable	Not applicable
Retrieve Dumbbell Over High Jump	Not applicable	Not applicable	Pass: 15/30	High Jump is part of Directed Jumping	High Jump is part of Directed Jumping
Broad Jump	Not applicable	Not applicable	Pass: 10/20	Broad Jump is part of Directed Jumping	Broad Jump is part of Directed Jumping
Seek Back Lost Article	Not applicable	Not applicable	Not applicable	Pass: 15/30	Pass: 15/30
Directed Jumping	Not applicable	Not applicable	Not applicable	Pass: 20/40	Pass: 20/40
Discriminate by Scent	Not applicable	Not applicable	Not applicable	Pass: 23/45	Pass: 23/45
Work by Signals	Not applicable	Not applicable	Not applicable	Pass: 15/30	Pass: 15/30
Speak, Refuse, or Retrieve	Not applicable	Not applicable	Not applicable	Pass: 10/20	Pass: 10/20
Number of passes required	One	Three	Three	Three	Five more UD Three first places
Total Score	Pass: 115/130	Pass: 170/200	Pass: 170/200	Pass: 170/200	Pass: 185/200

As the table shows, some tasks are identical in each class, some differ in their requirements and the potential points that could be scored in each class, and others are not applicable in a certain class.

After testing both approaches - general or specific templates - it seemed simpler to use specific templates, one template for each class. This provided tighter control when modifying task facts and made it simpler when checking and displaying the final result.

Pre-Competition Eligibility

Checking for pre-competition eligibility presented a minor design problem. A handler must be a member of the Royal NSW Canine Council to compete in any competitions run by the Council. A dog must be at least 6 months old to compete in a competition.

The problem was that the age check was required for both Encouragement Class and Novice Class. This is because Encouragement Class is not an official competition; rather it is an unofficial and optional competition to provide competition experience. However, it did not seem useful to check this twice, that is, once in each class. Therefore, it was included in the pre-competition check.

Dog Obedience Breed Advisor

Building the Dog Obedience Breed Advisor presented a number of problems.

Constructing the Sample Database

Constructing the sample database presented some issues rather than problems. Resolving them was important, however, because the quality of the information provided by the Dog Obedience Breed Advisor depends on the quality of the information in the sample database.

The main issue was how to determine a sensible sample of dog breeds, attributes, and values in the database, given that it was not practical to create a comprehensive database of all known purebreed and crossbreed dogs with all possible attributes and values.

An early decision was made to limit the sample database to approximately 50 breeds. This was thought to be a convenient manageable number that would also allow realistic searches. Therefore, the issue became how to select those breeds.

First, it was decided to include only purebreed dogs. If it were possible to include crossbreeds, it would be difficult to find reasons to select or reject appropriate breeds from the very large number of possible crossbreeds. Additionally, crossbreeds usually take on many of the characteristics of their purebreed parents, therefore, limiting the sample database to purebreeds was not seen as a significant issue.

Second, since the Advisor is designed to provide advice about obedience competitions and titles endorsed by the Royal New South Wales Canine Council, Australia [21], which is a member of the Australian National Kennel Council [2], it was decided to only include purebreeds recognised by the Australian National Kennel Council.

Third, since the Australian National Kennel Council recognises more than 180 breeds, it was decided to limit the sample database to the more popular dog breeds based on recent registration statistics compiled by the Australian National Kennel Council.

Fourth, it was decided to exclude variations of the same breed (for example, to include the Dachshund but not the Dachshund (Long Haired), Dachshund (Min. Long Haired), Dachshund (Min. Smooth Haired), Dachshund (Min. Wire Haired), Dachshund (Smooth Haired), or Dachshund (Wire Haired))

Fifth, it was decided to limit searchable attributes to those often regarded as important (based on this student's own experience plus advice from other dog owners) when distinguishing among dog breeds (size, coat length, grooming requirements, activity, temperament, nature, and tolerance for children, animals, and strangers).

Sixth, it was decided to include obedience information from the list compiled by Coren [5], based on his research into the intelligence of dogs. This ranks dogs according to breed-dependent working/obedience intelligence based on the breed's speed of understanding new commands and responsiveness to obey a new command when first given. This does not necessarily correlate to a specific dog's learning and problem solving ability or its intelligence.

Finally, it was decided to abbreviate some of the values used to describe searchable attributes (s for small, m for medium, l for large, x for extra large, y for yes, n for no, and so on). This would reduce the amount of information displayed on the screen and make it easier for a user to quickly enter a value.

After considering all criteria, the final sample database was created with 63 dog breeds, each having 14 attributes. Most have an obedience ranking, however, some popular breeds are unranked.

Building the Find Facilities

The "Find" facilities enable the user to quickly find a dog breed based on the unique breed attributes of Breed Name and Obedience Rank.

Note Obedience Rank is not really unique but it seemed more useful to have a quick Find option for a breed's obedience rank, rather than include it in a search option.

For the "Find a Breed" option, the main issue was whether to require exact matches or allow partial matches. For example, to require Beagle or to allow B* or B?? or Bea, and so on. It was decided to require exact matches. However, both uppercase and lowercase letters are acceptable, for example, beagle or BEAGLE or BeAgLe.

For the "Find Breed by Obedience Rank" option, the main issue was how to allow searching for breeds with no obedience ranking. The simplest technique was to assign zero (0) to unranked breeds and allow users to search for breeds with 0. This is the technique used.

Building the "Search by All Breed Attributes" Facility

Building the "Search by All Breed Attributes" facility presented some major problems, all of which were gradually overcome.

The intention of the revised project proposal was to include one comprehensive search facility in the Dog Obedience Breed Advisor in addition to the two "Find" facilities. This would ask a series of questions, one question for each non-unique breed attribute, to enable the user to specify an exact value or any value for each searchable attribute.

Searchable attributes are breed group, obedience group rating, breed size, coat length, grooming requirements, activity level, temperament, nature, tolerance for children, tolerance for other animals, and tolerance for strangers.

There were more than ten attributes, each having from three to seven exact values plus "any", allowing the user to search the sample database for a dog breed that matched any possible combination of these attributes and values.

The problem encountered was that, when a certain question was added to the series of questions (specifically the seventh of eleven questions, enquiring about Breed Temperament), CLIPS abended immediately after displaying the results of a search. It seemed that there were too many questions in the series and/or too many "any" answers to the questions.

Given that the comprehensive search facility and the means to specify an "any" response were regarded as high priority requirements, numerous approaches were taken to try and solve the problem.

First, the number of questions were reduced. It seemed that having an "any" response to a smaller number of questions was safe; having more caused CLIPS to abend. However, reducing the number of questions would mean that the search facility was no longer comprehensive.

Second, the number of records in the sample database were reduced. This did not seem to help. Reducing the number of records would also reduce the number of dog breeds for which obedience and other information was available, which would reduce the usefulness of the expert system.

Third, the rule that performed the search for matching records in the sample database was rewritten. The original rule looked like this:

```
(defrule BREED::phase-breed-go-display-skb-records-match
 (declare (salience 90))
 (phase breed-go-display-skb)
 (or ;if defrule activated from Find a Breed
     (breed $?breed)
     ;if defrule activated from Find a Rank
     (orank ?orank)
     ;if defrule activated from Search the Database
     (and
          (or (group ?group) (group z))
          (or (orat ?orat) (orat z))
          (or (size ?size) (size z))
          (or (coat ?coat) (coat z))
          (or (groom ?groom) (groom z))
          (or (activity ?activity) (activity z))
          (or (tempment ?tempment) (tempment z))
     );and
     ;if defrule activated from Display the Database
     (display all)
 );or
 ;match the dog(s)
 (dog (group ?group) (breed $?breed)
      (orat ?orat) (ogrp ?ogrp) (orank ?orank)
      (size ?size) (coat ?coat) (groom ?groom)
      (activity ?activity) (tempment ?tempment) (nature ?nature)
      (kidok ?kidok) (aniok ?aniok) (strok ?strok)
 );dog
=>
 (format t "%-11s | %-30s | %-9s | %-5d | %-4d | %-4s | %-4s | %-5s | %-8s | %-11s | %-6s | %-5s | %-6s | %-8s |
%-30s | %-11s%n"
          ?group (implode$ ?breed) ?orat ?ogrp ?orank ?size ?coat ?groom ?activity ?tempment ?nature ?kidok
?aniok ?strok (implode$ ?breed) ?group
 );format
);defrule
```

After rereading Chapter 11 in Giarratano and Riley [10] about efficiency in rule-based languages, the LHS pattern matching was arranged to the following, but this did not work:

```
...
 ;match the dog(s)
 (dog (group ?group) (breed $?breed)
      (orat ?orat) (ogrp ?ogrp) (orank ?orank)
      (size ?size) (coat ?coat) (groom ?groom)
      (activity ?activity) (tempment ?tempment) (nature ?nature)
      (kidok ?kidok) (aniok ?aniok) (strok ?strok)
 );dog
 (or ;if defrule activated from Find a Breed
     (breed $?breed)
     ;if defrule activated from Find a Rank
     (orank ?orank)
     ;if defrule activated from Search the Database
     (and
          (or (group ?group) (group z))
          (or (orat ?orat) (orat z))
          (or (size ?size) (size z))
          (or (coat ?coat) (coat z))
          (or (groom ?groom) (groom z))
          (or (activity ?activity) (activity z))
          (or (tempment ?tempment) (tempment z))
     );and
     ;if defrule activated from Display the Database
     (display all)
 );or
...
```

The LHS pattern matching was split into the following four separate rules but this did not work.

First rule:

```
...
 (dog (group ?group) (breed $?breed)
      (orat ?orat) (ogrp ?ogrp) (orank ?orank)
      (size ?size) (coat ?coat) (groom ?groom)
      (activity ?activity) (tempment ?tempment) (nature ?nature)
      (kidok ?kidok) (aniok ?aniok) (strok ?strok)
 )
 (breed $?breed)
...
```

Second rule:

```
...
 (dog (group ?group) (breed $?breed)
      (orat ?orat) (ogrp ?ogrp) (orank ?orank)
      (size ?size) (coat ?coat) (groom ?groom)
      (activity ?activity) (tempment ?tempment) (nature ?nature)
      (kidok ?kidok) (aniok ?aniok) (strok ?strok)
 )
 (orank ?orank)
...
```

Third rule:

```
...
```

```
 (dog (group ?group) (breed $?breed)
      (orat ?orat) (ogrp ?ogrp) (orank ?orank)
      (size ?size) (coat ?coat) (groom ?groom)
      (activity ?activity) (tempment ?tempment) (nature ?nature)
      (kidok ?kidok) (aniok ?aniok) (strok ?strok)
 )
 (or (group ?group) (group z))
 (or (orat ?orat) (orat z))
 (or (size ?size) (size z))
 (or (coat ?coat) (coat z))
 (or (groom ?groom) (groom z))
 (or (activity ?activity) (activity z))
 (or (tempment ?tempment) (tempment z))
...
```

Fourth rule:

```
...
 (dog (group ?group) (breed $?breed)
      (orat ?orat) (ogrp ?ogrp) (orank ?orank)
      (size ?size) (coat ?coat) (groom ?groom)
      (activity ?activity) (tempment ?tempment) (nature ?nature)
      (kidok ?kidok) (aniok ?aniok) (strok ?strok)
 )
 (display all)
...
```

Fourth, numerous attempts were made to work out another way of achieving the LHS logic that would also solve the problem but none were successful.

Fifth, an email [14] was sent to an expert on Expert Systems, Dr Michael Kirley at Charles Sturt University, for advice. He consulted some colleagues. He tried various techniques to solve the problem [12], including changing the Conflict Resolution Strategy from depth (CLIPS abended) to breadth (CLIPS did not abend), which suggested a problem in the rule logic. However, none of these strategies were successful.

Sixth, the idea of having one comprehensive search facility was abandoned. The search facility was split into major categories: obedience, physical features, behavioural attributes, and tolerance. This would allow a user to answer a smaller subset of questions within each category. This would generate fewer partial matches as the sample database is searched, which would mean that CLIPS would not abend. This technique worked. The following code segment shows the revised menu:

```
...
 (printout t crlf
  "----------------------------------------" crlf
  "2. Dog Obedience Breed Advisor Menu" crlf
  "----------------------------------------" crlf
  "The Dog Obedience Breed Advisor enables you to" crlf
  "obtain obedience information about various dogs breeds" crlf
  "using a Sample Dog Breed Database." crlf
  crlf
  "1. Find a Breed Group" crlf
  "2. Find a Breed" crlf
  "3. Find Breeds by Obedience Group Rating" crlf
  "4. Find Breed by Obedience Rank" crlf
  "5. Search by Physical Features" crlf
  "6. Search by Behavioural Attributes" crlf
  "7. Search by Tolerance" crlf
  "8. Display the Whole Database" crlf
  "0. Return to the MAIN MENU" crlf
  crlf
  "Please select an option (" ?min "-" ?max "): "
 );printout
...
```

Finally (reluctant to give up), a question was posted to the CLIPS Developers Board [15]. Soon after, an answer was received from Gary Riley, one of the developers of CLIPS, that provided a solution to the problem [20]. After testing, the solution was merged into the Dog Obedience Breed Advisor to provide the comprehensive search facility.

Final Observations

This section makes a few final observations, in no particular order, about the creation of the Dog Obedience Advisor.

Since I am interested in dogs in general and beagles in particular, choosing the broad knowledge domain for this project was not difficult. My original idea was to create a Dog Breed Selector. But there are numerous such selectors available online [for example, 3, 6, 7, 8, 18, 22, 23] and I did not want to duplicate the idea. However, to the best of my knowledge, there is no expert system available to check a handler and dog for eligibility and success in relation to various types of canine competitions. Hence, canine competitions seemed like a good choice for an expert system.

I gathered some ideas for how to present a menu from two of the example CLIPS programs distributed with Giarratano and Riley [10], in particular RX7.CLP and STOVE.CLP.

Many of the rules used by the Dog Obedience Breed Advisor use the IF ... THEN, SWITCH, and WHILE procedural programming language constructs. Often these are used for error checking after obtaining user input. Considerable thought was given to whether to perform error checking within the same rule used to obtain user input or whether to use separate rules for error checking. An early design decision was made to do as much error checking as possibly within the same rule to minimise the number of rules in the program. Giarratano and Riley [10, pp.498-499] regard this as an acceptable technique.

In addition to reasons already discussed, another reason for wanting to include the Dog Obedience Breed Advisor was to learn as many CLIPS language constructs as possible.

From a usefulness point of view, the Dog Obedience Advisor is probably not as useful as it could be. This is because the Advisor uses rules for obedience competitions run by the Royal NSW Canine Council, Australia. Other states and territories in Australia, and other countries, may not use the same rules. The Advisor is very specific. However, given that much of the literature on expert systems seems to suggest that focussed expert systems are better than more general expert systems, this is possibly not such a disadvantage.

Originally I hoped to create an expert system with a friendly graphical user interface so that users could use a pointing device, such as a mouse, to point and click when answering questions. I spent some time investigating alternatives to CLIPS, such as Jess, WebCLIPS, FuzzyCLIPS, wxCLIPS, ES-Builder Expert System Shell, and e2gLite Expert System Shell. In the end, I thought it was more important to learn the fundamentals of creating an expert system without the distraction of thinking about GUIs. Therefore, I tried to make the Dog Obedience Advisor as friendly as possible within the text-only limitations of CLIPS. I spent some time experimenting with the `format` construct to ensure that results from the Dog Obedience Breed Advisor were as well presented as possible.

Conclusion

This report has discussed the development of a Dog Obedience Advisor expert system. The Advisor has two parts: a Dog Obedience Competition Advisor; and a Dog Obedience Breed Advisor. The former is designed to check a handler and dog for eligibility and success in relation to various classes of canine Obedience competitions; the latter to provide obedience information about selected dog breeds contained in a sample database on which users can perform various types of searches.

The report summarised the original proposal for the development of a general Dog Competition Checker expert system. It explained the reasons for revising the project and building a specific Dog Obedience Advisor instead. It provided a general overview and detailed description of the Dog Obedience Competition Advisor and Dog Obedience Breed Advisor. It discussed some of the problems and issues encountered during the development of the Dog Obedience Advisor. To conclude, some final observations were made about the expert system.

References

1 *ANKC Obedience Rules* (online). (nd). http://safianski.customer.netspace.net.au/Training/1Rules/ObedienceRules.htm. [Accessed 8 Mar. 2003].

2 *Australian National Kennel Council* (online). (24 Jan. 2002). http://www.ankc.aust.com/default.html. [Accessed 8 Mar. 2003].

3 *Breed selector dogs* (online). (2003). iVillage Inc. http://www.ivillage.com/pets/tools/breedselector/questions/dog/size. [Accessed 8 Mar. 2003].

4 *Breed table* (online). (nd). about-dogs.com. http://www.about-dogs.com/breed_table.htm. [Accessed 8. Mar. 2003].

5 Coren, S. (1995). *The Intelligence of Dogs: a guide to the thoughts, emotions, and inner lives of our canine companions*. Bantam.

6 *Dog breed info center* (online). (2003). Green Bridge Station Technologies. http://www.dogbreedinfo.com/search.htm. [Accessed 8. Mar. 2003].

7 *Dog breed selector* (online). (2001). Discovery Communications. http://animal.discovery.com/guides/dogs/selector/selector1.jsp. [Accessed 8. Mar. 2003].

8 *Dog selector* (online). (2002). SelectSmart.com. http://selectsmart.com/DOG/. [Accessed 8. Mar. 2003].

9 Giarratano, J. (2002). *CLIPS User's Guide*. Version 6.20.

10 Giarratano, J. and Riley, G. (1998). *Expert systems: principles and programming*. 3rd ed. PWS Publishing: Boston, MA.

11 Kirley, M. (2003a). *Re: Slight change of focus?*. Personal email to J. New, dated 28 Apr. 2003.

12 Kirley, M. (2003b). *Re: CLIPS crashes occasionally*. Personal email to J. New, dated 6 May 2003.

13 New, J. (2003a). *Slight change of focus?*. Personal email to M. Kirley, dated 23 Apr. 2003.

14 New, J. (2003b). *CLIPS crashes occasionally*. Personal email to M. Kirley, dated 1 May 2003.

15 New, J. (2003c). Problem using "wildcards" to search a database. *CLIPS Developer's Board* (online). 5 May 2003. http://www.cpbinc.com/clips/viewtopic.php?t=1573. [Accessed 5 May 2003].

16 *Obedience* (online). (nd). http://www.rnswcc.org.au/compages/obedienc.html. Royal NSW Canine Council. [Accessed 8 Mar. 2003].

17 *Obedience trials in Australia* (online). (22 Feb. 1997). http://www.dogsites.com.au/internet_library/obedience_trials_in_australia.html. [Accessed 8 Mar. 2003].

18 *Quiz for choosing a dog* (online). (nd). about-dogs.com. http://www.about-dogs.com/quiz_for_choosing_dog.htm. [Accessed 8. Mar. 2003].

19 Riley, G. & Donnell, B. (2002). *CLIPS Reference Manual. Volume 1. Basic Programming Guide*. Version 6.20.

20 Riley, G. (2003). Problem using "wildcards" to search a database. *CLIPS Developer's Board* (online). 6 May 2003. http://www.cpbinc.com/clips/viewtopic.php?t=1573. [Accessed 6 May 2003].

21 *Royal NSW Canine Council* (online). (1 Apr. 2003). http://www.rnswcc.org.au/. [Accessed 8 Mar. 2003].

22 *Select a dog* (online). (2001). Waltham. http://www.waltham.com/dogs/select_a_dog.html. [Accessed 8. Mar. 2003].

23 *SelectaPet* (online). (1991). Petcare Information and Advisory Service. http://www.petnet.com.au/selectapet/dogselectapet.html. [Accessed 8. Mar. 2003].