An investigation into the use of a combined neural network and knowledge base system to classify types of glass and kinds of zoo animals

Andrew Peter Marlow

P2589383

(6,958 words)

November 12, 2003

Contents

List of Tables

List of Figures

Chapter 1

Summary

The zoo animal dataset was judged to be trivial and was easily classified with 100% success using a simple MLP of the form `(16,0,0,0,7)`.

The winning glass dataset network was an MLP that scored 81.3%. It was of the form `(9-6-8-0-7)` with overall learning coefficient 0.9, and coefficients for layers 1 and 2 of 0.6 and 0.4. It used the sigmoid transfer function and the delta-learn learning rule. The score of 81.3% was achieved with the default training run size (50,000). With a training run size of 400,000 it scored 86.9%.

It was found that MLPs could achieve higher scores by using a training run of a vast size and that RBFs could achieve high scores by a vast number of neurons. The increase in score was not dramatic though and was felt to have a high cost in terms of computing power required.

KBSs on their own did not fare any better than good MLPs and in some cases fared worse. However, classification results for the glass dataset were improved by combining an MLP with a KBS based on a C4.5 decision tree. The scores were similar to the high cost NNs but were achieved with simpler MLPs that ran very quickly. This shows that for non-trivial datasets is a beneficial to combine a NN with a KBS to get a high scoring system with relatively low processing overhead.

Chapter 2

Introduction

2.1 Artificial intelligence problems and approaches

2.1.1 Classification

This project is concerned with classification, i.e. predicting which group of like objects a selected object belongs to, given that the objects have a number of common attributes and given that the objects may form groups if arranged in a certain way.

Neural Networks (NNs) attempt to perform the classification using an approach that simulates the architecture of the brain. Knowledge based systems (KBSs) employ domain knowledge or rules extracted from the data itself.

This project explores the factors that govern the suitability of NNs or KBS for classifying objects from a particular dataset. Two datasets are used; one is a dataset of kinds of glass, the other is a set of zoo animals. Both sets are taken from the UCI Repository of Machine-learning Databases (Ref [2]). These datasets were chosen because of their radical differences. The attributes in the glass dataset appear to be poorly correlated whereas the attributes in the zoo animal dataset are highly correlated. It was thought that this difference might show that NNs are particularly suited for poorly correlated data and that domain knowledge can be used more effectively with highly correlated data. The glass dataset was also used as the data for a system that combines the NN and KBS approach to get the best of both worlds.

2.1.2 Neural networks

A NN consists of a number of nodes known as neurons, linked by weighted interconnections. The network as a whole has a number of inputs and a number of outputs. Each neuron has a number of inputs and a single output. The output of one neuron can form the input to another. The network is formed by an arrangement of these neurons and by the functions used to calculate each neuron's output. The output, given a set of inputs, is determined by a

relatively simple equation. The forms of equation and the kinds of coefficients determine what kind of NN we have. All are much simpler than the kinds of interconnection that occur in animal brains.

A perceptron is an arrangement of inputs and output for a single neuron. Perceptrons are arranged in layers, i.e. the inputs for a number of perceptrons comprise the total input for the arrangement and the outputs for all the perceptrons comprise the total output for the arrangement. An arrangement with just one such layer is known as a single layer perceptron, or SLP. More commonly, the outputs from one arrangment are feed as inputs to the next arrangement. The layers that feed their output as input to other layers are known as hidden layers. This arrangement of perceptrons is known as multi-layer perceptrons, or MLPs.

Another form of NN that is used in this study is the Kohonen network. Kohonen networks, sometimes referred to as self-organising maps or SOMs, have a layer known as the Kohonen layer which is an array of neurons fed by an input layer beneath. Neighbouring neurons that fire have their weightings adjusted to generate a higher output. This causes the network to learn to cluster together similar patterns.

A third form of NN used by this study is a Radial Basis Function (RBF) network. RBFs are simlar to MLPs except that the input neurons simply feed their input data into the nodes above and the equation that governs the output from neurons in the hidden layer describes a symmetrical function that can be thought of as a cluster in two dimensional pattern space or a hypershere is n-dimensional pattern space.

The knowledge representation held within a NN is held in a non-obvious form that is not open to inspection and that is not readily understood by humans. This inhibits the ability of the system to be validated, explained and refined. It resists analysis and is not associated with any domain knowledge.

2.1.3 Knowledge based systems

In a knowledge based system (KBS), the role of domain knowledge is separated from the software that controls the application of that knowledge. The domain knowledge is held in a module known as the *knowledge base* and the control module is known as the inference engine. The KBSs used on this project are rules based systems implemented in Prolog. The Prolog interpreter is the control module and the Prolog rules are the knowledge base. The separation of rules from control logic enables the domain knowledge to be clearly expressed.

When data is collected and examined, one can sometimes extract a general rule from the data, the logic is referred to as inductive. When using inductive logic to perform classification, domain knowledge is usually applied. There are other ways to arrive at a set of rules, including the use of specialist software to analyse patterns in the data. This was done on the project using a decision tree extractor known as C4.5 (Ref [8]).

2.2 Project description

Two datasets were chosen from the UCI Repository; the glass dataset and the zoo animals dataset. NN and KBS classifiers were developed for both datasets in order to show the effect that attribute correlation has on the effectiveness of the classifier and how complex the classifier needs to be. The glass dataset was used to refine the NN solution which was then integrated with the KBS solution in order to show the benefits of an integrated approach.

2.3 Project aims

The aims of the project are:

1. To develop NN and KBS classifiers for the glass and zoo animal datasets.
2. To refine the NN solution for the glass dataset by exploring the effectiveness of NNs that use back propagation (multi-layer perceptrons or MLPs), radial basis function (RBF) and self-organising map (Kohonen network). The aim is to find a network that achieves the highest score.
3. To demonstrate the relationship between attribute correlation and classification effectiveness.
4. To produce a classifier that combines NN and KBS techniques using Prolog and the NeuralWorks software supplied.

Only a limited amount of work seems to have been done on integrating NNs and KBSs. Generally, NNs are often used for classification in preference to KBSs, in spite of the disadvantages of NNs, mentioned earlier. There have been attempts to extract the knowledge acquired by a trained NN. For example, there is a software system called TREPAN (Ref [7]) which works directly on trained NNs but which requires that the attribute values are discrete. Another similar system is DecText (Ref [3]). Other systems include KBNN (Ref [1]) which incorporates case-based reasoning, and KBANN (Ref [5]) which involves injecting hand-crafted rules into a NN then retraining it with the benefits of the new rules. This project intends to show that dataset objects wrongly classified by a NN can be 'mopped up' by a KBS using a script that combines the two.

When this project was first started it was not clear how the NN/KBS integration would be done. Many of the ways covered by research papers were beyond my ability and beyond the scope of this project. However, the common theme seemed to be that one starts with one technique then uses the other to mop up any missed classifications. The project achieves that by starting with a NN, then using the Ross Quinlan C4.5 rules extraction program to generate a decision tree from which Prolog rules may be derived. These are used to produce a KBS classifier which is combined with the NN via a script that invokes the KBS as a 'mop-up' operation.

The challenge that remains to be solved is how to do this integration in a way that is completely independent of the dataset used. This was originally an aim of the project but was rejected during the course of the project as it was considered to be too hard. Very few studies seem to have tackled this. Ideally, what is needed is a generic technique that can be applied to any UCI dataset.

This study found that highly correlated data, such as is found in the zoo animals dataset, can be tackled either with a NN or a KBS and that integrated approach is not required. The integrated approach is beneficial for data that is poorly correlated. Hence, the integration in this project focused on just the glass dataset. Future work in this area would benefit from considering many datasets, all poorly correlated. With only one such dataset in this study, the integration approach did not need to be generic.

Chapter 3

Method and design

The classifier (either NN or KBS) must ask the user for an identifier for the chosen item (either a zoo animal or glass dataset id), then report which category the specified data belongs to. The NN classifiers were run using NeuralWorks then scored using `winner`. The KBS classifiers were Prolog programs run using Flex. A portable dialect of Prolog was used so that the code could also be run using either GNU Prolog or SWI-Prolog. This enabled the results to be checked automatically because the other Prolog interpreters lend themselves to scripting. It also enabled the KBS and NN techniques to be combined via a script. It was found that Flex did not lend itself to scripting. The script ran NeuralWorks first, then a Prolog program to 'mop up' any classification missed by the NN.

3.1 Neural networks

The NN classifiers used an input file consisting of records of normalised data attributes followed by expected output for the relevant category. After NeuralWorks has run the network, a results file is created which a suffix `.nnr`. The `winner` script is run on this file to apply the T396 scoring method and produce the number of successful matches as a number and a percentage.

This document refers to MLP NNs using the notation `(i,j,k,l,m)`, where `i` is the number of inputs, `m` is the number of outputs and `j`, `k` and `l` are the number of nodes in the hidden layers.

MLPs were created from the InstaNet->BackPropagation menu option of NeuralWorks. The learn rule was delta-rule, the sigmoid transfer function was used.

Kohonen networks were created from the InstaNet->SelfOrganisingMap menu option.

RBF (Radial Basis Function) networks were created from the InstaNet->RadialBasisFunctionNetwork menu option.

3.1.1 NN normalisation

The inputs are scaled between the minimum and maximum values to produce numbers in the range 0.0 to 1.0. The perl script used to do this is called mknni (it stands for make neural networks input) and is given in Appendix C, section C.3, page 58. It was used to produce the file `g1.nna` which was used for NeuralWorks runs.

3.1.2 NN Scoring

The NNs are scored according to *method 2* described in the study notes of T396 (Ref [11]). The scoring was calculated automatically, using a perl script that was developed as part of this project. The script is called *winner*, the listing is given in Appendix C (section C.4, page 60).

3.2 Knowledge-based Systems

The KBS classifier uses a set of backward-chaining rules to produce output that says which category the specified item belongs to.

The interactive Prolog required a portable way to obtain a string from the interactive user. This was done via the predicate `read_string`, shown in the section C.1.

3.2.1 KBS facts generation

The KBS classifiers used a set of Prolog fact statements followed by rules that were arrived at in various ways. The Prolog also contained the necessary statements to request input regarding the item to be classified and statements that reported on the category calculated by the rules. The fact statements were produced via perl scripts that processed the UCI data records. The scripts, `zoogen.pl` and `glassgen.pl` are given in Appendix C on pages 63 and 65.

3.2.2 Using C4.5

The C4.5 program was downloaded from
`http://www.cse.unsw.edu.au/ quinlan/c4.5r8.tar.gz`. The archive was decompressed and unpacked using `gzip` and `tar`. The `Makefile` was edited to use the GNU compiler (`gcc`) and to map the non-standard function `cfree` to the POSIX-standard `free`. C4.5 was built by typing the command `make` once these `Makefile` changes were done.

C4.5 is run using the command:

```
c4.5 -f<stem>
```

where `stem` is the first part of the data filename from the UCI repository. For example, to run C4.5 on the glass dataset the command

```
c4.5 -fglass
```

is used. This produces the binary output file `glass.tree`. An ASCII readable decision tree is printed to standard output.

Before C4.5 can be run, a file needs to be created that describes the domain range for each attribute. This data is in the file `<stem>.names`. These files exist in the UCI repository but are not always in the form required for C4.5. `glass.names` was created to run C4.5 on the glass dataset. The file is as shown in section C.2.

The files in the UCI repository contain `.names` files but these are not in the form required. The file above was created by hand by identifying the range for each attribute. It was simple in the case of the glass dataset because all the attributes are continuous, C4.5 also allows attribute types to be specified as consisting of discrete values.

3.3 The zoo dataset

The UCI data is in a file called `zoo.data`. The file consists of 101 records of comma separated fields. The first field is the animal name, the next 16 fields are the attributes and the final field is a type, which is a classification number in the range 1 to 7. The fields are shown in table A.1, page 23.

The task of the classifier is to produce the classification number given the animal name or id.

3.3.1 Neural networks

The `.nna` input file consists of 101 records, one per animal. Each record consists of the sixteen attributes followed by seven digits to encode the expected type. The type encoding is a simple binary code and is shown in table A.3.

The boolean attributes are coded as 0 for false and 1 for true. The data file was prepared from the UCI file using a simple perl script, `mkzoo`, (see section C.5, page 62) that strips off the animal name, converts the field delimiter from comma to space, and uses the type coding in the table above to produce each data record.

An MLP was created with 16 input nodes and 7 output nodes. There were no other layers, i.e. this was a `(16,0,0,0,7)` MLP. This is the simplest MLP given the number of inputs and outputs. 50,000 runs were used (in the run/learn option to NeuralWorks). The network achieved 100% classification success so no other networks were tried – they were not needed.

3.3.2 Knowledge based systems

The data was inspected manually to see which categories had certain attributes in common. For example, it was noticed that the combination

```
{hair=1, feathers=0, eggs=0, milk=1}
```

gave category 1 for most animals. This suggested that one way to arrive at suitable rules might be to find the maximum number of attributes in common for a given category. `zoo.data` was listed for certain category groups one at a time, with the records sorted by attributes so that like attributes would be grouped together. Using cygwin this was easy to do, using a combination of the `awk` and `sort` commands. Perl scripts were developed that reported the number of records that matched certain attribute settings. These record counts were matched against the number of records for the relevant category to see how accurate the rule was. The perl scripts used to report records that matched rules for categories 4, 5, 6 and 7 are `cat4.pl`, `cat5.pl`, `cat6.pl` and `cat7.pl` respectively and are listed in Appendix C. These scripts helped the development of rules that are used in the Prolog program `zoo01.pro`, listed in Appendix C, section C.8.5.

A refinement of `zoo01.pro` was created, `zoo02.pro`, which stored the category in each animal fact. This allowed the Prolog code to check to see if the results of a classification was correct and to report on successful matches. This enabled the effectiveness of the classification to be checked. A Perl script, `zoo02play.pl` was created (see section C.8.7), which creates a shellscript, `zoo02play` to run `zoo02.pro` with all the animal names supplied as input. This is in order to produce match reports for every animal in the dataset. The Prolog interpreter used for this automated run was SWI-Prolog, which lent itself to being automated in this way (Flex seems to be designed soley for interactive use).

The C4.5 program was used to produce a decision tree for the zoo animals. The tree produced was used to form rules in the Prolog program `zoo03.pro`, listed in Appendix C, section C.8.8.

The Prolog fact statements in `zoo01.pro` and `zoo03.pro` were generated using the `zoogen.pl` script. A modified form produced the fact statements for `zoo02.pro` by simply adding the category to the print statement.

3.4 The glass dataset

The glass dataset consists of 214 instances defined using 9 attributes. The UCI data is in a file called `glass.data`. The file consists of 214 records of comma separated fields. The first field is a glass id. This is an arbitrary number which is unique for each instance. The next 9 fields are the attributes. The final field is a type which is an integer in the range 1 to 7.

The fields are shown in table A.2, page 24.

3.4.1 Neural networks

MLPs were considered first. The number of attributes is 9 and there are 7 outputs so the networks were of the form `(9,x,y,z,7)`, where `x,y,z` are the number of neurons in hidden layers 1, 2 and 3 respectively. The number of neurons was varied and the RMS values records along with the number of matches

and percentage score. The RMS values reported after a 'Learn' run and a 'Test' run were recorded. These are referred to in the results as RMS_{learn} and RMS_{test} respectively. `x` was varied first. The network was performed best was selected for the next stage, varying the value of `y`. Similarly, the best network for values of `x` and `y` was selected to determine the best value for `z`. As the values of `x,y,z` were varied, the changes to the RMS values were used to guide the way the neuron counts were varied. For example, the RMS_{test} values for MLP `(9-6-y-0-7)` reduced only slightly with `y` in the range 9 to 12, changing from 0.2451 to 0.2472, indicating that further increases in the value of `y` would probably not yield better performance.

Kohonen networks were tried next. The sigmoid transfer function was used and a coordinate layer. Experiments were started with 7 neurons in the hidden layer, i.e. the network was of the form `(9,9,x,7,7)`. `x` was varied and the RMS values and scored recorded. The number of training runs was increased to see if this could improve on the scores obtained.

Finally, RBF networks were tried. The number of neurons in the hidden layer was set to 0, thus the form of the network was `(9,x,0,7)`. The number of neurons was increased until a near perfect score was achieved. Indications were that a perfect score might be achieved if the number of neurons could be increased still further but the limit supported by NeuralWorks was reached.

A winning network was decided on, based on the score achieved and the network complexity required to achieve it. The winner was an MLP. This MLP was then refined by altering the learning coefficients. The number of iterations in the learn/run cycle was varied to see if that would improve network performance. The winning configuration was noted for use in the integrated approach. It was decided that the overall MLP winner would use the default training run size (50,000). This is because of the time taken for larger training run sizes.

3.4.2 Knowledge based systems

It was very difficult to see any pattern to the data, unlike the zoo animals dataset. Several attempts were made to find overlaying attribute regions using Excel displays and checks made via scripts. The conclusion was reached that the data was too poorly correlated to submit to the same kind of analysis that was possible with the zoo animals.

The `glassgen.pl` script was used to generate Prolog facts about the normalised glass data. The data was normalised because it was felt it might be easier to deal with attribute overlaps in a similar way if the same range was used by each. A Prolog program, `glass01.pro` was used as an attempt to take the same sort of approach as for the zoo animals. `glass01.pro` was based on the observation that category 7 glass was characterised chiefly by the range of refractive index and iron content. The listing of `glass01.pro` is given in Appendix C, section C.9.1. Further data examination along these lines led to the conclusion that further rules would make very fine distinctions between data values indeed and that it was time to use C4.5.

The program `glass02.pro` is based on a C4.5 decision tree processing

`glass.data`. It was felt that there was no need to normalise the data in this case. The listing of `glass02.pro` is given in Appendix C, section C.9.2. This listing does not represent the complete tree. The complete Prolog program was developed in stages to ensure that as the rules built up the expected number of dataset entities were categorised correctly. This helped to avoid errors since the the tree produced by C4.5 seemed quite large giving scope for errors to creep in. The final program is `glass05.pro` and is given in Appendix C, section C.9.3. The MS-Windows program `Edge Diagrammer` was used to draw diagrams of C4.5 decision trees. These were found to be helpful in translating the tree into the equivalent Prolog program, as nodes could be checked off the tree once there was a corresponding rule for them.

`glass05.pro` as developed as an interactive program. A slight variant was created, `glass06.pro`, designed for batch mode so that queries could be made for every glass item in the dataset. This enables a score to be obtained for the KBS system. A listing of `glass06.pro` is given in Appendix C, section C.9.4. A Perl script, `glass06play.pl` was created (similar to `zoo02play`), which creates a shellscript, `glass06play` to run `glass06.pro` with all the glass ids supplied as input. This is in order to produce match reports for every glass item in the dataset.

3.5 A combined approach

The winning NN, a `(9-6-8-0-7)` MLP (see sections 3.4.1 and 4.2.1) was used as a starting point. Two versions of this NN were used, one before the learning coefficient optimisations, and one after. These NeuralWorks configurations were saved as `g1.nnd` and `g2.nnd` respectively.

A script was developed, `marry.pl` (so called because it attempts to marry the use of an NN and KBS), `marry.pl` is shown in Appendix C, section C.10.

`marry.pl` uses the `automate` program program that comes with NeuralWorks. `automate` is used to run NeuralWorks without user interaction. The command `ounw -xuautomate` needs to be executed. The `.nnr` output file is deleted before running `automate`. Also, `marry.pl` re-creates the input file to `automate` on each run, because `automate` alters the input file during the run. The input file to `automate` is called `automate.dat`. It refers the MLP that was selected as the best NN. This was saved to a file named as `g1.nnd`. Thus the `automate.dat` file looked like this (for the `g1` case):

```
open g1.nnd
initialize
set learnfile g1.nna
learn 50000
test
```

After running NeuralWorks, `marry.pl` applies the `winner` scoring method to the output file produced by `automate` and reports on whether or not the classification was correct. If the result is wrong then the script runs a modified form

of `glass06.pro` using SWI-Prolog (Flex was not used because I was unable to find out how to run Flex via a script). The Prolog used is slightly modified from `glass06.pro` because:

1. There were technical problems doing a completely automated run of the Prolog program. It proved to be difficult to get the glass id into the system just by running `glass06.pro` via the script, so eventually the script created a prolog program on the fly based on a template, `glass06b.pro`. The created program contained the required glass id in the `go` predicate.

2. The modified program did not bother to echo the attributes. The attribute echoing was primarily of use for the interactive testing and during debugging. Thus, `glass06b.pro` is the same as `glass06.pro` except that the attribute trace statements are removed and the `go` predicate is removed.

Chapter 4

Results

Tables and figures, mainly for NN results, are in Appendix A. Program output, such as Prolog output and NeuralWorks `nnr` files, is in Appendix B.

4.1 The zoo dataset

4.1.1 Neural networks

See section B.1, page 32 for the output of NeuralWorks when the MLP is used. The file was named `z1.nnr`. It shows that the classification is straightfoward, as the relevant output column is very close to unity whilst the other columns remain close to zero every time. The RMS value was 0.0261. The network scores 100%. This result was not expected when the project first started. It was anticipated that some exploration of neural networks would be required before the highest scoring network could be found. As it turns out, there was no need to consider anything other than a very simple MLP.

4.1.2 Knowledge based systems

`zoo01.pro` was designed as a interactive program. It was run several times for a number of animals and produced the correct results in each case.

`zoo2play` was used to run `zoo02.pro` on every animal, producing the output in section B.2, page 34. It shows all animals except for the seasnake correctly classified. The output has been massaged slightly by removing blank lines in order to make the listing shorter.

C4.5 was run on the zoo animals data, even though the simple results show that this step was not required. This was done out of curiosity, to see how simple the decision tree would be. The C4.5 output is shown in Appendix B, section B.3. It shows that C4.5 did not even need to prune the tree.

4.2 The glass dataset

4.2.1 Neural networks

The tables and figures for these results are given in Appendix A.

MLPs

Tables A.4, A.5 and A.6 show the variation in RMS values and scores when the number of neurons is varied in the hidden layers. From these tables and associated barcharts it can be seen that the `(9-6-8-0-7)` MLP gives the best results. This network was used to explore the effect of varying the learning coefficients.

Results for varying the main learning coefficient and the learning coefficient for hidden layer 1 are shown in tables A.7 and A.8. Table A.7 shows that a main coefficient value of 0.9 gave best results. This value was used when varying the learning coefficient value for the first hidden layer. Table A.8 shows that 0.3 and 0.6 yield the best values but 0.3 is the default value so it would appear that this cannot be improved upon. Table A.9 shows the variation in RMS and score values when varying learning coefficient 3. The best score was acheived with a coefficient value of 0.4. Finally, table A.10 shows the variation in number of matches and score when the size of the training run is varied. The best results were achieved with a training run size of 400 thousand. This gave a score of 86.9%. However, this network took a very long time to run. The winning MLP with a default training run size was used when doing the combined approach.

Kohonen networks

The Kohonen network results are shown in table A.11. They are much poorer than the MLP results. Increasing the number of training runs seems to make little difference as shown by the results in table A.12.

RBFs

The form of the network was `(9-x-0-7)`. Table A.13 and the associated bar chart A.4 shows the results when `x` is varied, for a training run size of 20 thousand. Since a score in excess of 99% was achieved, further refinement of the RBF approach was deemed unnecessary.

4.2.2 Knowledge based systems

`glass02.pro` to `glass05.pro` were designed as interactive programs. `glass05.pro` was run several times for a number of glass instances and produced the correct results in each case.

The output from the C4.5 run on `glass.data` is shown in section B.4, page 40. The decision tree which was used to form the rules in `glass05.pro` is

shown in figure A.5, page 31. Note that to keep the size of the diagram down the attribute names are shown as chemical symbols (e.g Al for aluminium, Si for silicon etc). Refractive Index is shown as RI.

`glass06.pro`, which uses the same classification rules as `glass05`, produced the output shown in section B.5. There are 155 matches out of 214, i.e. a score of 72.4%.

4.3 The Combined Approach

A fragment of the output is shown in section B.6. The output for runs `g1.nna` and `g2.nna` were captured in files so that the number of matches could conveniently be counted. `g1.nna` gave 187 matches ((87.4%) of which 169 matches (79.0%) were from the NN. `g2.nna` also gave 187 matches ((87.4%) of which 186 matches (87.0%) were from the NN.

Chapter 5

Discussion of results

5.1 The zoo dataset

5.1.1 Neural networks

A simple MLP gave perfect results because the variables are highly correlated. Producing correlation figures for any UCI dataset attributes is a challenge due to the number of attributes and the skills in numerical analysis required. It is therefore beyond the scope of this report to produce such figures to substantiate this claim. However, the fact that Prolog rules could be so easily arrived at by grouping the data sorting on various attributes, and that C4.5 did not need to prune the tree, shows that the data is highly correlated.

5.1.2 Knowledge based systems

The rules required were simple because the relationships between the variables is simple and amendable to manual inspection and consequent rule formation.

5.2 The glass dataset

5.2.1 Neural networks

MLPs gave good results, were simple and ran quickly. The Kohonen networks performed so poorly compared to the MLPs there seemed no point in refining them. Whilst an RBF-based network achieved a near perfect score the number of neurons was near the limit supported by the NeuralWorks software. The network took a significant amount of time to run with this many nodes. It was felt that this made the MLP the better solution.

MLPs

As the RMS value decreased, the scores tended to increase, as expected. Even a simple MP such as `(9-1-0-0-7)` achieves a respectable score (67.8%) but it is necessary to add more neurons and use hidden layers, to improve the result. The best MLP found, `(9-6-8-0-7)`, was improved still further by tweaking the learning coefficient values slightly, and increasing the size of the learning run. It was thought that this approach would only yield a small improvement but the score was 73.4% before tweaking and 86.9% after tweaking, a significant gain. This same score was achieved using an RBF but at a cost of an additional 136 neurons.

Kohonen networks

Table A.11 shows that a `(9-9-4-7-7)` Kohonen network performs roughly as well as a `(9-3-0-07)` MLP but it was easier to improve on the MLP performance. The `(9-9-4-7-7)` Kohonen network was already the peak performer and even drastic increases in the run size only gave a marginal improvement, from 70.1% to 72.0%.

From a literature search it seems as if Kohonen networks have not received much attention in recent years for classification problems. This is despite them being quoted as being good classifiers in general. This study found that their performance is unremarkable. Other studies seem to rate their performance as merely comparable to MLPs.

RBFs

The barchart in figure A.4 shows that although the score achieved by an RBF increases as the number of neurons increases, the rate of increase decays exponentially. It would seem that with enough neurons the RBF solution would give the perfect classifier in terms of score but the network would be huge. The tradeoff between score and network size is an area for further work (see section 7).

When the number of neurons in the RBF NN is at a similar level to the best performing MLP, the scores achieved are similar. However, the MLP can be siginificantly improved by tweaking other parameters whereas the RBF can only be improved by adding more neurons and the number that need to be added to get the corresponding improvement to a tweaked MLP increases exponentially. This suggests a certain inefficiency in RBFs. It would be interesting to see if this effect could be observed in classification problems using other UCI datasets.

The decision to pick an MLP as the best performing network in spite of what was achieved with RBFs was because MLPs are more efficient, i.e. a small quick MLP produces the same score as a large slow RBF.

5.2.2 Knowledge based systems

The KBS score for the glass dataset was a disappointment. However, it was not completely unexpected as the data seemed to be very hard to group by any understandable criterea. This seems to confirm that NNs are more suited to classification than KBSs. For example, glass id 73 which is of category 2, has the following record from `z1.nnr`, showing a correct classification:

```
0 1 0 0 0 0 0 0.008003 0.975664 0.050262 0.000178 0.000256 0 0.016695
```

whereas `glass05.pro` incorrectly classifies it as belonging to category 3.

Once C4.5 was used to form the rules the KBS lost some of the traditional advantages of KBSs, i.e. the ability for the rules to be easily understandable and for the rules to map onto the real world modelling of the problem. Tests on the ranges for certain attributes are repeated in nodes further down the decision tree, but with different specific values. There is no use of domain knowledge here. It is purely a device to find correlations between data items grouped in a particular way, as determined by the C4.5 algorithms. This seems to be a consequence of using a decision tree on poorly correlated data. This in turn suggests that a better, more easily understandable KBS might be possible if better algorithms could be found than those used in C4.5. Ross Quinlan, the author of C4.5 is currently working in this area and has produced a prototype, C5, which, it is claimed, performs better than C4.5. An evaluation copy of C5 was run on the glass dataset. It gave better classification results but not by much and that the decision tree was even more complex.

The Prolog code was much longer and harder to understand than the output produced by C4.5. The C4.5 output for the pruned tree is a mere 44 lines but the Prolog rules comprise 191 lines, excluding blank lines and trace statements. This suggests that it might be worthwhile to find a way to generate the Prolog rules from the decision tree.

It is interesting to note that with poorly correlated data such as is found in the glass dataset, a NN performs better than a decision tree based KBS, provided a sufficiently large number of neurons is used. This suggests that poor data correlation is at the heart of the problem and that it is solved either by the brute force approach of adding more neurons to an RBF, or by improving on the algorithms that find correlation when producing a decision tree.

5.3 The Combined Approach

The KBS mopping up operation appears to be a success. The output shows that some of the items that the NN fails to classify correctly are classified correctly by the KBS. The overall performance of the combined approach is comparable to the best MLP that was arrived at by careful tuning.

Chapter 6

Conclusion

6.1 Trivial datasets can easily be classified with MLPs

The degree of data correlation was found to dominate the results found in this study. This makes the choice of datasets crucial. It was not realised when the study first began how trivial the zoo animal dataset really is. Even so, despite the traditional advantages a KBS affords over a NN, the MLP gave perfect results and the KBS did not. This study concludes that NNs give best classification results for trivial datasets.

6.2 There are dangers in using just one dataset

One of the reasons the zoo animals was chosen as well as the glass dataset was to avoid the dangers in drawing conclusions from just one dataset. This is despite the fact that other NN performance evaluations also tend to use just a single dataset (see Ref [6] and Ref [9]). It was a pity that the zoo animals dataset did not contribute to the study in the way that was hoped. It would be interesting to do a study on multiple UCI datasets where the datasets are selected on the basis of poor data correlation.

Few studies appear to have been done using several non-trivial datasets from the UCI repository. Most studies seem to concentrate on a single set; one or two studies were found that used a few datasets. As examples, Ref [12] uses the house-votes, ionosphere, letter and OCR datasets; Ref [4] uses the iris, Ljubljan cancer and appendicitus datasets; Ref [10] uses breast cancer and character recognition datasets.

6.3 High-scoring NNs are expensive

The RBF NN gave the highest scores but at a considerable cost in network complexity and time to execute. The winning MLP also had its score boosted by dramatically increasing the size of training run but this too had a high run time cost.

6.4 The benefit of a NN/KBS combination

This study concludes that it is beneficial to combine the NN and KBS approaches to achieve better classification. The classification is better because it makes more efficient use of computer resources by lightening the machine load compared to a more CPU intensive run with a more complex NN.

Chapter 7

Suggestions for further work

7.1 MLP and RBF performance studes

MLPs have been compared with RBFs in order to tune RBF performance (see Ref [6] and Ref [9]) but this tends to be done with a single dataset. What is needed is a study with multiple datasets.

7.2 Attribute correlation software

There seem to be very few software packages available to perform the numerical analysis required to arrive at correlation matrices for multivariate data. The few that exist are commercial heavyweight packages such as SPSS and SAS and could therefore not be used on this project. An Open Source package could be developed that would enable correlation figures to be found and verified for all UCI datasets (currently there is only a limited amount of correlation data and no easy way to verify it).

7.3 C4.5 and Prolog

It was a suprise to find that the `.names` file in the UCI repository are not in the form required by C4.5. Also, it was tedious to create Prolog programs by hand from the C4.5 output. Future AI studies using the UCI datasets would benefit from having a `.names` file for each dataset that it is the form that C4.5 requires. It would also be useful to have a modified form of C4.5 that produces Prolog rules directly. Judging from the current state of C5 (C4.5's successor), this seems to be what Ross Quinlan has in mind. These steps would enable the entire UCI repository to have a KBS solution.

7.4 Automated NN/KBS integration for the UCI Repository

If the UCI repository had C4.5 `.names` files for each dataset, and there was a version of C4.5 that produced the Prolog rules automatically, then a script could be developed that could produce a KBS system for each dataset. Then another script could be produced, similar to `marry.pl` that runs a NN and mops up results using the generated KBS. The creation of the NN could also be automated, using similar techniques to those used in this study. For example, a script could be developed that would automate the normalisation and would work out a number of trial MLPs to use, selecting that which performed best according to the `winner` script.

This would automate the NN/KBS classification of the entire repository. This is probably the most important piece of future work as far as this project is concerned because of the limitations of using only two datasets. This would answer the question once and for all as to whether or not the integrated approach in this study was worthwhile in general.

Appendix A

Tables and Figures

Table A.1: Zoo dataset attributes

	Attribute	Type
animal name	unique for each instance	
1	hair	boolean
2	feathers	boolean
3	eggs	boolean
4	milk	boolean
5	airborne	boolean
6	aquatic	boolean
7	predator	boolean
8	toothed	boolean
9	backbone	boolean
10	breathes	boolean
11	venomous	boolean
12	fins	boolean
13	legs	integer
14	tail	boolean
15	domestic	boolean
16	catsize	boolean
	type	a classification number in the range 1 to 7

Table A.2: Glass dataset attributes

	Attribute
id	unique for each instance
1	Refractive index
2	sodium content
3	magnesium content
4	aluminium content
5	silcon content
6	potassium content
7	calcium content
8	barium content
9	iron content
type	1 building windows, float processed 2 build windows, non-float processed 3 vehicle windows, float processed 4 vehicle windows, non-float processed (none in this dataset) 5 containers 6 tableware 7 headlamps

Table A.3: Zoo animal category in Neural Network form

seven digits	type
1 0 0 0 0 0 0	1
0 1 0 0 0 0 0	2
0 0 1 0 0 0 0	3
0 0 0 1 0 0 0	4
0 0 0 0 1 0 0	5
0 0 0 0 0 1 0	6
0 0 0 0 0 0 1	7

Table A.4: Varying the neurons in H1

PE-H1-H2-H3-Out	RMS_{learn}	RMS_{test}	matches	% score
9-1-0-0-7	0.2927	0.2925	145	67.8
9-2-0-0-7	0.2946	0.2739	148	69.2
9-3-0-0-7	0.2609	0.2600	149	69.6
9-4-0-0-7	0.2545	0.2543	153	71.5
9-5-0-0-7	0.2595	0.2587	142	66.4
9-6-0-0-7	0.2540	0.2528	155	72.4
9-7-0-0-7	0.2512	0.2503	148	69.2
9-8-0-0-7	0.2527	0.2517	153	71.5
9-9-0-0-7	0.2488	0.2479	153	71.5
9-10-0-0-7	0.2509	0.2509	152	71.0
9-11-0-0-7	0.2488	0.2480	155	72.4
9-12-0-0-7	0.2472	0.2458	153	71.5

Figure A.1: Barchart of data from table A.4

Table A.5: Varying the neurons in H2

PE-H1-H2-H3-Out	RMS_{learn}	RMS_{test}	matches	% score
9-6-1-0-7	0.2875	0.2875	135	63.0
9-6-2-0-7	0.2607	0.2588	149	69.6
9-6-3-0-7	0.2581	0.2572	147	68.7
9-6-4-0-7	0.2543	0.2522	154	72.0
9-6-5-0-7	0.2511	0.2515	154	72.0
9-6-6-0-7	0.2505	0.2506	154	72.0
9-6-7-0-7	0.2517	0.2508	154	72.0
9-6-8-0-7	0.2433	0.2419	157	73.4
9-6-9-0-7	0.2463	0.2451	154	72.0
9-6-10-0-7	0.2526	0.2511	150	70.1
9-6-11-0-7	0.2463	0.2456	154	72.0
9-6-12-0-7	0.2475	0.2472	151	70.6

Figure A.2: Barchart of data from table A.5

Table A.6: Varying the neurons in H3

PE-H1-H2-H3-Out	RMS_{learn}	RMS_{test}	matches	% score
9-6-8-1-7	0.2926	0.2921	107	50.0
9-6-8-2-7	0.2609	0.2592	142	66.4
9-6-8-3-7	0.2616	0.2667	139	65.0
9-6-8-4-7	0.2640	0.2617	150	70.1
9-6-8-5-7	0.2598	0.2578	147	68.7
9-6-8-6-7	0.2469	0.2465	156	72.9
9-6-8-7-7	0.2533	0.2537	147	68.7
9-6-8-8-7	0.2546	0.2608	143	66.8
9-6-8-9-7	0.2531	0.2499	152	71.0
9-6-8-10-7	0.2542	0.2509	155	72.4
9-6-8-11-7	0.2487	0.2465	154	72.0
9-6-8-12-7	0.2526	0.2517	150	70.1

Figure A.3: Barchart of data from table A.6

Table A.7: Varying the main learning coefficient ratio

Coefficient	RMS_{learn}	RMS_{test}	matches	% score
0.0	0.2680	0.2680	141	66.4
0.1	0.2606	0.2606	148	69.2
0.2	0.2566	0.2565	150	70.1
0.3	0.2522	0.2518	154	72.0
0.4	0.2477	0.2470	155	72.4
0.5	0.2433	0.2419	157	73.4
0.6	0.2394	0.2372	160	74.8
0.7	0.2362	0.2332	163	76.2
0.8	0.2309	0.2260	165	77.1
0.9	0.2239	0.2160	171	79.9
0.95	0.2217	0.2134	171	79.9
0.955	0.2217	0.2129	170	79.4
1.0	0.2206	0.2217	161	75.2

Table A.8: Varying the learning coefficient ratio for H1

Coefficient	RMS_{learn}	RMS_{test}	matches	% score
0.0	0.3001	0.2966	108	50.5
0.1	0.2311	0.2282	160	74.8
0.2	0.2239	0.2162	167	78.0
0.3	0.2239	0.2160	171	79.9
0.4	0.2177	0.2120	168	78.5
0.5	0.2160	0.2102	170	79.4
0.6	0.2161	0.2099	171	79.9
0.7	0.2169	0.2144	166	77.6
0.8	0.2157	0.2087	170	79.4
0.9	0.2170	0.2094	168	78.5
1.0	0.2243	0.2258	155	72.4

Table A.9: Varying the learning coefficient ratio for H2

Coefficient	RMS_{learn}	RMS_{test}	matches	% score
0.0	0.2647	0.2626	145	67.8
0.1	0.2220	0.2140	168	78.5
0.4	0.2125	0.2059	174	81.3
1.0	0.2334	0.2272	160	74.8

Table A.10: Varying the size of the training run (MLP)

Run (thousands)	matches	% score
50	174	81.3
100	178	83.2
200	184	86.0
400	186	86.9
800	164	76.6

Table A.11: Varying `x` in a `(9,9,x,7,7)` Kohonen network

x	RMS_{learn}	RMS_{test}	matches	% score
1	0.3050	0.3018	113	52.8
2	0.2842	0.2627	125	58.4
3	0.2689	0.2672	143	66.8
4	0.2573	0.2558	150	70.1
5	0.2611	0.2599	148	69.2
6	0.2615	0.2601	145	67.8
7	0.2678	0.2664	144	67.3
8	0.2637	0.2623	144	67.3
9	0.2688	0.2686	140	65.4

Table A.12: Varying the size of the training run (Kohonen)

Run (thousands)	matches	% score
25	150	70.1
50	150	70.1
100	152	71.0
250	154	72.0
500	154	72.0

Table A.13: Varying the number of neurons in an RBF network

x	RMS_{learn}	RMS_{test}	matches	% score
1	0.3254	0.3251	99	46.3
2	0.3252	0.3256	99	46.3
3	0.2904	0.2894	115	53.7
4	0.2884	0.2880	117	54.7
5	0.2883	0.2865	121	56.5
6	0.2776	0.2762	132	61.7
7	0.2761	0.2751	131	61.2
8	0.2742	0.2725	137	64.0
9	0.2729	0.2711	136	63.6
10	0.2698	0.2683	138	64.5
20	0.2561	0.2539	157	73.4
40	0.2265	0.2252	168	78.5
80	0.2053	0.2032	179	83.6
100	0.1972	0.1954	178	83.2
150	0.1791	0.1781	186	86.9
200	0.1681	0.1667	192	89.7
250	0.1556	0.1539	199	93.0
300	0.1445	0.1432	204	95.3
350	0.1350	0.1338	205	95.8
400	0.1164	0.1154	210	98.1
450	0.1086	0.1077	212	99.1

Figure A.4: Barchart of data from table A.13

Figure A.5: Decision tree for `glass05.pro`

Appendix B

Output files

B.1 z1.nnr

The file `z1.nnr` is a NeuralWorks output file. Tabs have been converted to single space and the first 7 fields have been converted from floating point to integer since the value is either 0 or 1. This makes the file easier to read.

```
1 0 0 0 0 0 0 0.999398 0.000098 0.000416 0.000000 0.004571 0.000334 0.000076
1 0 0 0 0 0 0 0.999378 0.000381 0.010674 0.000000 0.000718 0.000193 0.000000
0 0 0 1 0 0 0 0.001709 0.002525 0.012161 0.979942 0.008884 0.000000 0.002558
1 0 0 0 0 0 0 0.999398 0.000098 0.000416 0.000000 0.004571 0.000334 0.000076
1 0 0 0 0 0 0 0.999259 0.000377 0.005518 0.000000 0.000369 0.000011 0.000039
1 0 0 0 0 0 0 0.999378 0.000381 0.010674 0.000000 0.000718 0.000193 0.000000
1 0 0 0 0 0 0 0.999470 0.000387 0.003909 0.000000 0.000296 0.000140 0.000000
0 0 0 1 0 0 0 0.002389 0.002594 0.008630 0.997278 0.007138 0.000000 0.000065
0 0 0 1 0 0 0 0.001709 0.002525 0.012161 0.979942 0.008884 0.000000 0.002558
1 0 0 0 0 0 0 0.997444 0.000066 0.000226 0.000000 0.024946 0.019436 0.000000
1 0 0 0 0 0 0 0.999259 0.000377 0.005518 0.000000 0.000369 0.000011 0.000039
0 1 0 0 0 0 0 0.000596 0.995212 0.003298 0.000584 0.000297 0.003805 0.000008
0 0 0 1 0 0 0 0.001709 0.002525 0.012161 0.979942 0.008884 0.000000 0.002558
0 0 0 0 0 0 1 0.004162 0.029876 0.031498 0.060796 0.000691 0.002665 0.996020
0 0 0 0 0 0 1 0.000868 0.001615 0.000190 0.000183 0.031935 0.004569 0.985283
0 0 0 0 0 0 1 0.000512 0.000433 0.000091 0.000006 0.044580 0.029092 0.954502
0 1 0 0 0 0 0 0.000426 0.995081 0.004657 0.000078 0.000371 0.000291 0.000309
1 0 0 0 0 0 0 0.999378 0.000381 0.010674 0.000000 0.000718 0.000193 0.000000
0 0 0 1 0 0 0 0.010077 0.003856 0.015773 0.986285 0.001289 0.000000 0.003389
1 0 0 0 0 0 0 0.983776 0.003767 0.000366 0.035457 0.004438 0.000000 0.001092
0 1 0 0 0 0 0 0.000596 0.995212 0.003298 0.000584 0.000297 0.003805 0.000008
0 1 0 0 0 0 0 0.000303 0.993364 0.000234 0.001300 0.016937 0.000212 0.000038
1 0 0 0 0 0 0 0.999378 0.000381 0.010674 0.000000 0.000718 0.000193 0.000000
0 1 0 0 0 0 0 0.003009 0.996815 0.011705 0.000828 0.000104 0.001160 0.000018
0 0 0 0 0 1 0 0.001823 0.000675 0.004643 0.000000 0.045475 0.941696 0.149831
```

```
0 0 0 0 1 0 0 0.002276 0.000226 0.009342 0.000017 0.973816 0.000088 0.001181
0 0 0 0 1 0 0 0.000597 0.000092 0.034830 0.000000 0.921407 0.000029 0.009076
1 0 0 0 0 0 0 0.993097 0.007125 0.001994 0.000035 0.000736 0.001763 0.000000
1 0 0 0 0 0 0 0.999378 0.000381 0.010674 0.000000 0.000718 0.000193 0.000000
1 0 0 0 0 0 0 0.999697 0.000370 0.000317 0.000000 0.001334 0.000037 0.000137
0 0 0 0 0 1 0 0.000573 0.005188 0.000537 0.000000 0.009836 0.995341 0.004239
1 0 0 0 0 0 0 0.999470 0.000387 0.003909 0.000000 0.000296 0.000140 0.000000
1 0 0 0 0 0 0 0.999702 0.000368 0.001691 0.000031 0.006288 0.000922 0.000011
0 1 0 0 0 0 0 0.000255 0.993289 0.000120 0.000180 0.008762 0.000012 0.000842
0 0 0 1 0 0 0 0.002035 0.002554 0.023376 0.997175 0.017171 0.000000 0.000114
1 0 0 0 0 0 0 0.996857 0.000253 0.003005 0.000000 0.002051 0.000633 0.000000
1 0 0 0 0 0 0 0.996311 0.000249 0.008220 0.000000 0.004969 0.000871 0.000000
0 1 0 0 0 0 0 0.000426 0.995081 0.004657 0.000078 0.000371 0.000291 0.000309
0 0 0 1 0 0 0 0.001709 0.002525 0.012161 0.979942 0.008884 0.000000 0.002558
0 0 0 0 0 1 0 0.003859 0.000516 0.000019 0.000000 0.000095 0.987261 0.003923
0 0 0 0 0 1 0 0.012438 0.001242 0.000014 0.000000 0.000734 0.996930 0.000897
0 1 0 0 0 0 0 0.001355 0.963256 0.039064 0.000121 0.001775 0.000022 0.012643
0 0 0 0 0 1 0 0.000481 0.005130 0.000276 0.000000 0.005071 0.922054 0.087186
0 1 0 0 0 0 0 0.000507 0.995137 0.009015 0.000563 0.000722 0.005227 0.000014
1 0 0 0 0 0 0 0.999259 0.000377 0.005518 0.000000 0.000369 0.000011 0.000039
1 0 0 0 0 0 0 0.999259 0.000377 0.005518 0.000000 0.000369 0.000011 0.000039
0 0 0 0 0 0 1 0.000512 0.000433 0.000091 0.000006 0.044580 0.029092 0.954502
1 0 0 0 0 0 0 0.999259 0.000377 0.005518 0.000000 0.000369 0.000011 0.000039
1 0 0 0 0 0 0 0.998761 0.000276 0.000143 0.000000 0.008714 0.000000 0.000107
1 0 0 0 0 0 0 0.995609 0.000246 0.004244 0.000000 0.002555 0.000048 0.000030
1 0 0 0 0 0 0 0.999259 0.000377 0.005518 0.000000 0.000369 0.000011 0.000039
0 0 0 0 0 1 0 0.012438 0.001242 0.000014 0.000000 0.000734 0.996930 0.000897
0 0 0 0 1 0 0 0.001850 0.000872 0.111778 0.000044 0.749190 0.000000 0.000613
0 0 0 0 0 0 1 0.001792 0.000178 0.000056 0.000000 0.009411 0.041432 0.897085
1 0 0 0 0 0 0 0.995609 0.000246 0.004244 0.000000 0.002555 0.000048 0.000030
1 0 0 0 0 0 0 0.999378 0.000381 0.010674 0.000000 0.000718 0.000193 0.000000
0 1 0 0 0 0 0 0.009520 0.975934 0.093298 0.001287 0.000499 0.000088 0.000756
0 1 0 0 0 0 0 0.000596 0.995212 0.003298 0.000584 0.000297 0.003805 0.000008
0 1 0 0 0 0 0 0.004799 0.967029 0.001358 0.000412 0.006065 0.000000 0.044206
0 1 0 0 0 0 0 0.000507 0.995137 0.009015 0.000563 0.000722 0.005227 0.000014
0 0 0 1 0 0 0 0.010077 0.003856 0.015773 0.986285 0.001289 0.000000 0.003389
0 0 0 1 0 0 0 0.001709 0.002525 0.012161 0.979942 0.008884 0.000000 0.002558
0 0 1 0 0 0 0 0.002332 0.006740 0.987992 0.003212 0.019362 0.000000 0.017442
1 0 0 0 0 0 0 0.943893 0.002722 0.000024 0.000000 0.000169 0.000000 0.007676
1 0 0 0 0 0 0 0.999259 0.000377 0.005518 0.000000 0.000369 0.000011 0.000039
1 0 0 0 0 0 0 0.999470 0.000387 0.003909 0.000000 0.000296 0.000140 0.000000
1 0 0 0 0 0 0 0.983776 0.003767 0.000366 0.035457 0.004438 0.000000 0.001092
1 0 0 0 0 0 0 0.999259 0.000377 0.005518 0.000000 0.000369 0.000011 0.000039
1 0 0 0 0 0 0 0.999369 0.000383 0.002014 0.000000 0.000152 0.000008 0.000022
1 0 0 0 0 0 0 0.999259 0.000377 0.005518 0.000000 0.000369 0.000011 0.000039
1 0 0 0 0 0 0 0.999470 0.000387 0.003909 0.000000 0.000296 0.000140 0.000000
```

```
0 1 0 0 0 0 0 0.008003 0.975664 0.050262 0.000178 0.000256 0.000000 0.016695
0 0 0 0 0 0 1 0.008077 0.000774 0.092030 0.000000 0.002768 0.065863 0.848191
0 0 0 1 0 0 0 0.002035 0.002554 0.023376 0.997175 0.017171 0.000000 0.000114
1 0 0 0 0 0 0 0.999390 0.000234 0.000000 0.008708 0.004085 0.000000 0.000444
1 0 0 0 0 0 0 0.998728 0.000242 0.000000 0.000804 0.000466 0.000000 0.000072
0 0 1 0 0 0 0 0.031982 0.011940 0.843788 0.067754 0.106767 0.000000 0.095658
0 0 0 0 0 0 1 0.000654 0.009120 0.003187 0.029850 0.005171 0.000035 0.999811
0 1 0 0 0 0 0 0.000255 0.993289 0.000120 0.000180 0.008762 0.000012 0.000842
0 1 0 0 0 0 0 0.000255 0.993289 0.000120 0.000180 0.008762 0.000012 0.000842
0 0 1 0 0 0 0 0.008844 0.016339 0.955556 0.015431 0.058945 0.000000 0.002287
0 0 0 0 0 0 1 0.008833 0.033931 0.041125 0.010644 0.016557 0.054871 0.851356
0 0 0 1 0 0 0 0.002035 0.002554 0.023376 0.997175 0.017171 0.000000 0.000114
0 1 0 0 0 0 0 0.000507 0.995137 0.009015 0.000563 0.000722 0.005227 0.000014
1 0 0 0 0 0 0 0.997822 0.000929 0.017066 0.000055 0.003518 0.000134 0.000000
0 0 0 0 0 0 1 0.000666 0.000837 0.000131 0.000034 0.037752 0.011592 0.974011
0 0 0 1 0 0 0 0.002659 0.001579 0.057786 0.936654 0.000407 0.000000 0.025659
0 1 0 0 0 0 0 0.001801 0.995651 0.000304 0.001913 0.002474 0.000047 0.000050
0 0 0 0 0 1 0 0.001823 0.000675 0.004643 0.000000 0.045475 0.941696 0.149831
0 0 0 0 1 0 0 0.002710 0.000229 0.018005 0.000122 0.986392 0.001588 0.000053
0 0 1 0 0 0 0 0.019045 0.047746 0.781978 0.000076 0.002140 0.001247 0.001076
0 0 1 0 0 0 0 0.003092 0.001191 0.830469 0.000019 0.111361 0.000070 0.000225
0 0 0 1 0 0 0 0.010077 0.003856 0.015773 0.986285 0.001289 0.000000 0.003389
1 0 0 0 0 0 0 0.993097 0.007125 0.001994 0.000035 0.000736 0.001763 0.000000
1 0 0 0 0 0 0 0.996311 0.000249 0.008220 0.000000 0.004969 0.000871 0.000000
0 1 0 0 0 0 0 0.002527 0.996778 0.006054 0.000115 0.000053 0.000064 0.000410
1 0 0 0 0 0 0 0.999633 0.001420 0.022104 0.000081 0.000508 0.000030 0.000000
0 0 0 0 0 1 0 0.003288 0.000508 0.000053 0.000000 0.000231 0.990708 0.006905
1 0 0 0 0 0 0 0.999259 0.000377 0.005518 0.000000 0.000369 0.000011 0.000039
0 0 0 0 0 0 1 0.008833 0.033931 0.041125 0.010644 0.016557 0.054871 0.851356
0 1 0 0 0 0 0 0.000507 0.995137 0.009015 0.000563 0.000722 0.005227 0.000014
```

B.2 zoo02playout

Blank lines have been removed to make the listing slghtly shorter.

```
Welcome to SWI-Prolog (Version 5.1.5)
Copyright (c) 1990-2002 University of Amsterdam.
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to redistribute it under certain conditions.
Please visit http://www.swi-prolog.org for details.
For help, use ?- help(Topic). or ?- apropos(Word).
1 ?- Warning: (/home/marlowa/laptop/everything/andrew/ou/tm426/software/zoo02.pro:109):
        Singleton variables: [Cat, Match, Animal]
Warning: (/home/marlowa/laptop/everything/andrew/ou/tm426/software/zoo02.pro:139):
        Singleton variables: [Airborne, Predator, Legs, Tail, Domestic, Catsize]
Warning: (/home/marlowa/laptop/everything/andrew/ou/tm426/software/zoo02.pro:152):
```

```
        Singleton variables: [Hair, Airborne, Aquatic, Predator, Legs, Tail, Domestic, C
Warning: (/home/marlowa/laptop/everything/andrew/ou/tm426/software/zoo02.pro:161):
        Singleton variables: [Predator, Toothed, Breathes, Venomous, Legs, Catsize]
Warning: (/home/marlowa/laptop/everything/andrew/ou/tm426/software/zoo02.pro:170):
        Singleton variables: [Predator, Venomous, Tail, Domestic, Catsize, Tails]
Warning: (/home/marlowa/laptop/everything/andrew/ou/tm426/software/zoo02.pro:179):
        Singleton variables: [Predator, Breathes, Venomous, Tail, Domestic, Catsize]
Warning: (/home/marlowa/laptop/everything/andrew/ou/tm426/software/zoo02.pro:188):
        Singleton variables: [Hair, Airborne, Predator, Venomous, Tail, Domestic, Catsiz
Warning: (/home/marlowa/laptop/everything/andrew/ou/tm426/software/zoo02.pro:197):
        Singleton variables: [Eggs, Aquatic, Breathes, Venomous, Tail, Catsize]
Warning: (/home/marlowa/laptop/everything/andrew/ou/tm426/software/zoo02.pro:225):
        Singleton variables: [Cat, Category]
% zoo02.pro compiled 0.00 sec, 29,240 bytes
Yes
Enter animal name: aardvark category is 1 match = 1
Yes
Enter animal name: antelope category is 1 match = 1
Yes
Enter animal name: bass category is 4 match = 1
Yes
Enter animal name: bear category is 1 match = 1
Yes
Enter animal name: boar category is 1 match = 1
Yes
Enter animal name: buffalo category is 1 match = 1
Yes
Enter animal name: calf category is 1 match = 1
Yes
Enter animal name: carp category is 4 match = 1
Yes
Enter animal name: catfish category is 4 match = 1
Yes
Enter animal name: cavy category is 1 match = 1
Yes
Enter animal name: cheetah category is 1 match = 1
Yes
Enter animal name: chicken category is 2 match = 1
Yes
Enter animal name: chub category is 4 match = 1
Yes
Enter animal name: clam category is 7 match = 1
Yes
Enter animal name: crab category is 7 match = 1
Yes
Enter animal name: crayfish category is 7 match = 1
```

```
Yes
Enter animal name: crow category is 2 match = 1
Yes
Enter animal name: deer category is 1 match = 1
Yes
Enter animal name: dogfish category is 4 match = 1
Yes
Enter animal name: dolphin category is 1 match = 1
Yes
Enter animal name: dove category is 2 match = 1
Yes
Enter animal name: duck category is 2 match = 1
Yes
Enter animal name: elephant category is 1 match = 1
Yes
Enter animal name: flamingo category is 2 match = 1
Yes
Enter animal name: flea category is 6 match = 1
Yes
Enter animal name: frog category is 5 match = 1
Yes
Enter animal name: frog category is 5 match = 1
Yes
Enter animal name: fruitbat category is 1 match = 1
Yes
Enter animal name: giraffe category is 1 match = 1
Yes
Enter animal name: girl category is 1 match = 1
Yes
Enter animal name: gnat category is 6 match = 1
Yes
Enter animal name: goat category is 1 match = 1
Yes
Enter animal name: gorilla category is 1 match = 1
Yes
Enter animal name: gull category is 2 match = 1
Yes
Enter animal name: haddock category is 4 match = 1
Yes
Enter animal name: hamster category is 1 match = 1
Yes
Enter animal name: hare category is 1 match = 1
Yes
Enter animal name: hawk category is 2 match = 1
Yes
Enter animal name: herring category is 4 match = 1
```

```
Yes
Enter animal name: honeybee category is 6 match = 1
Yes
Enter animal name: housefly category is 6 match = 1
Yes
Enter animal name: kiwi category is 2 match = 1
Yes
Enter animal name: ladybird category is 6 match = 1
Yes
Enter animal name: lark category is 2 match = 1
Yes
Enter animal name: leopard category is 1 match = 1
Yes
Enter animal name: lion category is 1 match = 1
Yes
Enter animal name: lobster category is 7 match = 1
Yes
Enter animal name: lynx category is 1 match = 1
Yes
Enter animal name: mink category is 1 match = 1
Yes
Enter animal name: mole category is 1 match = 1
Yes
Enter animal name: mongoose category is 1 match = 1
Yes
Enter animal name: moth category is 6 match = 1
Yes
Enter animal name: newt category is 5 match = 1
Yes
Enter animal name: octopus category is 7 match = 1
Yes
Enter animal name: opossum category is 1 match = 1
Yes
Enter animal name: oryx category is 1 match = 1
Yes
Enter animal name: ostrich category is 2 match = 1
Yes
Enter animal name: parakeet category is 2 match = 1
Yes
Enter animal name: penguin category is 2 match = 1
Yes
Enter animal name: pheasant category is 2 match = 1
Yes
Enter animal name: pike category is 4 match = 1
Yes
Enter animal name: piranha category is 4 match = 1
```

```
Yes
Enter animal name: pitviper category is 3 match = 1
Yes
Enter animal name: platypus category is 1 match = 1
Yes
Enter animal name: polecat category is 1 match = 1
Yes
Enter animal name: pony category is 1 match = 1
Yes
Enter animal name: porpoise category is 1 match = 1
Yes
Enter animal name: puma category is 1 match = 1
Yes
Enter animal name: pussycat category is 1 match = 1
Yes
Enter animal name: raccoon category is 1 match = 1
Yes
Enter animal name: reindeer category is 1 match = 1
Yes
Enter animal name: rhea category is 2 match = 1
Yes
Enter animal name: scorpion category is 7 match = 1
Yes
Enter animal name: seahorse category is 4 match = 1
Yes
Enter animal name: seal category is 1 match = 1
Yes
Enter animal name: sealion category is 1 match = 1
Yes
Enter animal name:
No
Enter animal name: seawasp category is 7 match = 1
Yes
Enter animal name: skimmer category is 2 match = 1
Yes
Enter animal name: skua category is 2 match = 1
Yes
Enter animal name: slowworm category is 3 match = 1
Yes
Enter animal name: slug category is 7 match = 1
Yes
Enter animal name: sole category is 4 match = 1
Yes
Enter animal name: sparrow category is 2 match = 1
Yes
Enter animal name: squirrel category is 1 match = 1
```

```
Yes
Enter animal name: starfish category is 7 match = 1
Yes
Enter animal name: stingray category is 4 match = 1
Yes
Enter animal name: swan category is 2 match = 1
Yes
Enter animal name: termite category is 6 match = 1
Yes
Enter animal name: toad category is 5 match = 1
Yes
Enter animal name: tortoise category is 3 match = 1
Yes
Enter animal name: tuatara category is 3 match = 1
Yes
Enter animal name: tuna category is 4 match = 1
Yes
Enter animal name: vampire category is 1 match = 1
Yes
Enter animal name: vole category is 1 match = 1
Yes
Enter animal name: vulture category is 2 match = 1
Yes
Enter animal name: wallaby category is 1 match = 1
Yes
Enter animal name: wasp category is 6 match = 1
Yes
Enter animal name: wolf category is 1 match = 1
Yes
Enter animal name: worm category is 7 match = 1
Yes
Enter animal name: wren category is 2 match = 1
Yes
103 ?-
% halt
```

B.3 C4.5 output from the zoo animals

```
C4.5 [release 8] decision tree generator        Sun Nov  2 12:02:23 2003
---------------------------------------

    Options:
        File stem <zoo>

Read 101 cases (17 attributes) from zoo.data
```

```
Decision Tree:

milk = 1: 1 (41.0)
milk = 0:
|   fins = 1: 4 (13.0)
|   fins = 0:
|   |   feathers = 1: 2 (20.0)
|   |   feathers = 0:
|   |   |   backbone = 0:
|   |   |   |   airborne = 1: 6 (6.0)
|   |   |   |   airborne = 0:
|   |   |   |   |   predator = 1: 7 (8.0)
|   |   |   |   |   predator = 0:
|   |   |   |   |   |   legs = 0: 7 (2.0)
|   |   |   |   |   |   legs = 2: 6 (0.0)
|   |   |   |   |   |   legs = 4: 6 (0.0)
|   |   |   |   |   |   legs = 5: 6 (0.0)
|   |   |   |   |   |   legs = 6: 6 (2.0)
|   |   |   |   |   |   legs = 8: 6 (0.0)
|   |   |   backbone = 1:
|   |   |   |   tail = 0: 5 (3.0)
|   |   |   |   tail = 1: 3 (6.0/1.0)


Tree saved


Evaluation on training data (101 items):

         Before Pruning           After Pruning
        ----------------   ---------------------------
        Size      Errors   Size      Errors   Estimate

          21    1( 1.0%)     21    1( 1.0%)    (11.8%)   <<
```

B.4 C4.5 output from the glass dataset

This is the output produced by running C4.5 on the glass dataset.

```
C4.5 [release 8] decision tree generator        Sun Oct 26 11:40:35 2003
----------------------------------------

    Options:
        File stem <glass>
```

```
Read 214 cases (10 attributes) from glass.data

Decision Tree:

Ba <= 0.27 :
|   Mg <= 2.41 :
|   |   K <= 0.03 :
|   |   |   Na <= 13.75 : 2 (3.0)
|   |   |   Na > 13.75 : 6 (9.0)
|   |   K > 0.03 :
|   |   |   Na > 13.49 : 2 (7.0/1.0)
|   |   |   Na <= 13.49 :
|   |   |   |   RI <= 1.5241 : 5 (13.0/1.0)
|   |   |   |   RI > 1.5241 : 2 (3.0)
|   Mg > 2.41 :
|   |   Al <= 1.41 :
|   |   |   RI <= 1.51707 :
|   |   |   |   RI <= 1.51596 : 1 (3.0)
|   |   |   |   RI > 1.51596 :
|   |   |   |   |   Fe > 0.12 : 2 (2.0)
|   |   |   |   |   Fe <= 0.12 :
|   |   |   |   |   |   Al > 1.27 : 3 (5.0)
|   |   |   |   |   |   Al <= 1.27 :
|   |   |   |   |   |   |   Mg <= 3.47 : 3 (2.0)
|   |   |   |   |   |   |   Mg > 3.47 : 2 (2.0)
|   |   |   RI > 1.51707 :
|   |   |   |   K <= 0.23 :
|   |   |   |   |   Mg <= 3.34 : 2 (2.0)
|   |   |   |   |   Mg > 3.34 :
|   |   |   |   |   |   Si > 72.64 : 3 (3.0)
|   |   |   |   |   |   Si <= 72.64 :
|   |   |   |   |   |   |   Na <= 14.01 : 1 (14.0)
|   |   |   |   |   |   |   Na > 14.01 :
|   |   |   |   |   |   |   |   Al <= 0.51 : 1 (3.0)
|   |   |   |   |   |   |   |   Al > 0.51 :
|   |   |   |   |   |   |   |   |   Si <= 71.36 : 1 (3.0/1.0)
|   |   |   |   |   |   |   |   |   Si > 71.36 : 3 (2.0)
|   |   |   |   K > 0.23 :
|   |   |   |   |   Mg > 3.75 : 2 (10.0)
|   |   |   |   |   Mg <= 3.75 :
|   |   |   |   |   |   Fe <= 0.14 :
|   |   |   |   |   |   |   RI <= 1.52043 : 1 (36.0)
|   |   |   |   |   |   |   RI > 1.52043 : 2 (2.0/1.0)
|   |   |   |   |   |   Fe > 0.14 :
|   |   |   |   |   |   |   Al <= 1.17 : 2 (5.0)
|   |   |   |   |   |   |   Al > 1.17 : 1 (6.0/1.0)
```

```
|   |   Al > 1.41 :
|   |   |   Si > 72.49 : 2 (39.0/6.0)
|   |   |   Si <= 72.49 :
|   |   |   |   Ca <= 8.28 : 2 (6.0)
|   |   |   |   Ca > 8.28 : 3 (5.0/1.0)
Ba > 0.27 :
|   Si <= 70.16 : 2 (2.0/1.0)
|   Si > 70.16 : 7 (27.0/1.0)


Simplified Decision Tree:

Ba <= 0.27 :
|   Mg <= 2.41 :
|   |   K <= 0.03 :
|   |   |   Na <= 13.75 : 2 (3.0/1.1)
|   |   |   Na > 13.75 : 6 (9.0/1.3)
|   |   K > 0.03 :
|   |   |   Na > 13.49 : 2 (7.0/2.4)
|   |   |   Na <= 13.49 :
|   |   |   |   RI <= 1.5241 : 5 (13.0/2.5)
|   |   |   |   RI > 1.5241 : 2 (3.0/1.1)
|   Mg > 2.41 :
|   |   Al <= 1.41 :
|   |   |   RI <= 1.51707 :
|   |   |   |   RI <= 1.51596 : 1 (3.0/1.1)
|   |   |   |   RI > 1.51596 :
|   |   |   |   |   Fe > 0.12 : 2 (2.0/1.0)
|   |   |   |   |   Fe <= 0.12 :
|   |   |   |   |   |   Al > 1.27 : 3 (5.0/1.2)
|   |   |   |   |   |   Al <= 1.27 :
|   |   |   |   |   |   |   Mg <= 3.47 : 3 (2.0/1.0)
|   |   |   |   |   |   |   Mg > 3.47 : 2 (2.0/1.0)
|   |   |   RI > 1.51707 :
|   |   |   |   K <= 0.23 :
|   |   |   |   |   Mg <= 3.34 : 2 (2.0/1.0)
|   |   |   |   |   Mg > 3.34 :
|   |   |   |   |   |   Si <= 72.64 : 1 (22.0/4.8)
|   |   |   |   |   |   Si > 72.64 : 3 (3.0/1.1)
|   |   |   |   K > 0.23 :
|   |   |   |   |   Mg > 3.75 : 2 (10.0/1.3)
|   |   |   |   |   Mg <= 3.75 :
|   |   |   |   |   |   Fe <= 0.14 :
|   |   |   |   |   |   |   RI <= 1.52043 : 1 (36.0/1.4)
|   |   |   |   |   |   |   RI > 1.52043 : 2 (2.0/1.8)
|   |   |   |   |   |   Fe > 0.14 :
```

```
|   |   |   |   |   |   |   Al <= 1.17 : 2 (5.0/1.2)
|   |   |   |   |   |   |   Al > 1.17 : 1 (6.0/2.3)
|   |   Al > 1.41 :
|   |   |   Si > 72.49 : 2 (39.0/8.3)
|   |   |   Si <= 72.49 :
|   |   |   |   Ca <= 8.28 : 2 (6.0/1.2)
|   |   |   |   Ca > 8.28 : 3 (5.0/2.3)
Ba > 0.27 :
|   Si <= 70.16 : 2 (2.0/1.8)
|   Si > 70.16 : 7 (27.0/2.6)


Tree saved


Evaluation on training data (214 items):

         Before Pruning           After Pruning
         ----------------   ---------------------------
         Size      Errors   Size      Errors   Estimate

           51   14( 6.5%)     45   16( 7.5%)    (20.9%)   <<
```

B.5 glass06.pro output

Blank lines have been removed to make the listing slghtly shorter.

```
Welcome to SWI-Prolog (Version 5.1.5)
Copyright (c) 1990-2002 University of Amsterdam.
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to redistribute it under certain conditions.
Please visit http://www.swi-prolog.org for details.
For help, use ?- help(Topic). or ?- apropos(Word).
1 ?- Warning: (/home/marlowa/laptop/everything/andrew/ou/tm426/software/glass06.pro:223)
        Singleton variables: [Cat, Match, Gid]
Warning: (/home/marlowa/laptop/everything/andrew/ou/tm426/software/glass06.pro:269):
        Singleton variables: [RefractiveIndex, Sodium, Magnesium, Aluminium, Potassium,
Warning: (/home/marlowa/laptop/everything/andrew/ou/tm426/software/glass06.pro:275):
        Singleton variables: [RefractiveIndex, Sodium, Magnesium, Aluminium, Potassium,
Warning: (/home/marlowa/laptop/everything/andrew/ou/tm426/software/glass06.pro:309):
        Singleton variables: [RefractiveIndex, Magnesium, Aluminium, Silicon, Potassium,
Warning: (/home/marlowa/laptop/everything/andrew/ou/tm426/software/glass06.pro:315):
        Singleton variables: [RefractiveIndex, Magnesium, Aluminium, Silicon, Potassium,
Warning: (/home/marlowa/laptop/everything/andrew/ou/tm426/software/glass06.pro:328):
        Singleton variables: [RefractiveIndex, Magnesium, Aluminium, Silicon, Potassium,
Warning: (/home/marlowa/laptop/everything/andrew/ou/tm426/software/glass06.pro:334):
```

```
        Singleton variables: [Sodium, Magnesium, Aluminium, Silicon, Potassium, Calcium,
Warning: (/home/marlowa/laptop/everything/andrew/ou/tm426/software/glass06.pro:340):
        Singleton variables: [Sodium, Magnesium, Aluminium, Silicon, Potassium, Calcium,
Warning: (/home/marlowa/laptop/everything/andrew/ou/tm426/software/glass06.pro:374):
        Singleton variables: [Sodium, Magnesium, Aluminium, Silicon, Potassium, Calcium,
Warning: (/home/marlowa/laptop/everything/andrew/ou/tm426/software/glass06.pro:394):
        Singleton variables: [RefractiveIndex, Sodium, Magnesium, Aluminium, Silicon, Po
Warning: (/home/marlowa/laptop/everything/andrew/ou/tm426/software/glass06.pro:408):
        Singleton variables: [RefractiveIndex, Sodium, Magnesium, Silicon, Potassium, Ca
Warning: (/home/marlowa/laptop/everything/andrew/ou/tm426/software/glass06.pro:414):
        Singleton variables: [RefractiveIndex, Sodium, Aluminium, Silicon, Potassium, Ca
Warning: (/home/marlowa/laptop/everything/andrew/ou/tm426/software/glass06.pro:420):
        Singleton variables: [RefractiveIndex, Sodium, Aluminium, Silicon, Potassium, Ca
Warning: (/home/marlowa/laptop/everything/andrew/ou/tm426/software/glass06.pro:440):
        Singleton variables: [RefractiveIndex, Sodium, Aluminium, Silicon, Potassium, Ca
Warning: (/home/marlowa/laptop/everything/andrew/ou/tm426/software/glass06.pro:453):
        Singleton variables: [RefractiveIndex, Sodium, Magnesium, Aluminium, Potassium,
Warning: (/home/marlowa/laptop/everything/andrew/ou/tm426/software/glass06.pro:459):
        Singleton variables: [RefractiveIndex, Sodium, Magnesium, Aluminium, Potassium,
Warning: (/home/marlowa/laptop/everything/andrew/ou/tm426/software/glass06.pro:472):
        Singleton variables: [RefractiveIndex, Sodium, Aluminium, Silicon, Potassium, Ca
Warning: (/home/marlowa/laptop/everything/andrew/ou/tm426/software/glass06.pro:492):
        Singleton variables: [Sodium, Magnesium, Aluminium, Silicon, Potassium, Calcium,
Warning: (/home/marlowa/laptop/everything/andrew/ou/tm426/software/glass06.pro:498):
        Singleton variables: [Sodium, Magnesium, Aluminium, Silicon, Potassium, Calcium,
Warning: (/home/marlowa/laptop/everything/andrew/ou/tm426/software/glass06.pro:504):
        Singleton variables: [RefractiveIndex, Sodium, Magnesium, Silicon, Potassium, Ca
Warning: (/home/marlowa/laptop/everything/andrew/ou/tm426/software/glass06.pro:510):
        Singleton variables: [RefractiveIndex, Sodium, Magnesium, Silicon, Potassium, Ca
Warning: (/home/marlowa/laptop/everything/andrew/ou/tm426/software/glass06.pro:516):
        Clauses of test2Si/10 are not together in the source-file
Warning: (/home/marlowa/laptop/everything/andrew/ou/tm426/software/glass06.pro:523):
        Singleton variables: [RefractiveIndex, Sodium, Magnesium, Aluminium, Potassium,
Warning: (/home/marlowa/laptop/everything/andrew/ou/tm426/software/glass06.pro:529):
        Singleton variables: [RefractiveIndex, Sodium, Magnesium, Aluminium, Silicon, Po
Warning: (/home/marlowa/laptop/everything/andrew/ou/tm426/software/glass06.pro:535):
        Singleton variables: [RefractiveIndex, Sodium, Magnesium, Aluminium, Silicon, Po
Warning: (/home/marlowa/laptop/everything/andrew/ou/tm426/software/glass06.pro:546):
        Singleton variables: [Cat, Category]
% glass06.pro compiled 0.02 sec, 48,252 bytes
Yes
Enter glass id: id 1 match = 1
Yes
Enter glass id: id 2 match = 1
Yes
Enter glass id: id 3 match = 0
```

```
Yes
Enter glass id: id 4 match = 1
Yes
Enter glass id: id 5 match = 1
Yes
Enter glass id: id 6 match = 0
Yes
Enter glass id: id 7 match = 1
Yes
Enter glass id: id 8 match = 1
Yes
Enter glass id: id 9 match = 1
Yes
Enter glass id: id 10 match = 1
Yes
Enter glass id: id 11 match = 0
Yes
Enter glass id: id 12 match = 1
Yes
Enter glass id: id 13 match = 1
Yes
Enter glass id: id 14 match = 1
Yes
Enter glass id: id 15 match = 1
Yes
Enter glass id: id 16 match = 1
Yes
Enter glass id: id 17 match = 1
Yes
Enter glass id: id 18 match = 1
Yes
Enter glass id: id 19 match = 1
Yes
Enter glass id: id 20 match = 0
Yes
Enter glass id: id 21 match = 0
Yes
Enter glass id: id 22 match = 1
Yes
Enter glass id: id 23 match = 1
Yes
Enter glass id: id 24 match = 1
Yes
Enter glass id: id 25 match = 1
Yes
Enter glass id: id 26 match = 1
```

```
Yes
Enter glass id: id 27 match = 1
Yes
Enter glass id: id 28 match = 1
Yes
Enter glass id: id 29 match = 0
Yes
Enter glass id: id 30 match = 1
Yes
Enter glass id: id 31 match = 1
Yes
Enter glass id: id 32 match = 1
Yes
Enter glass id: id 33 match = 1
Yes
Enter glass id: id 34 match = 1
Yes
Enter glass id: id 35 match = 1
Yes
Enter glass id: id 36 match = 1
Yes
Enter glass id: id 37 match = 1
Yes
Enter glass id: id 38 match = 1
Yes
Enter glass id: id 39 match = 1
Yes
Enter glass id: id 40 match = 1
Yes
Enter glass id: id 41 match = 1
Yes
Enter glass id: id 42 match = 1
Yes
Enter glass id: id 43 match = 1
Yes
Enter glass id: id 44 match = 1
Yes
Enter glass id: id 45 match = 1
Yes
Enter glass id: id 46 match = 1
Yes
Enter glass id: id 47 match = 1
Yes
Enter glass id: id 48 match = 1
Yes
Enter glass id: id 49 match = 1
```

```
Yes
Enter glass id: id 50 match = 1
Yes
Enter glass id: id 51 match = 1
Yes
Enter glass id: id 52 match = 1
Yes
Enter glass id: id 53 match = 1
Yes
Enter glass id: id 54 match = 1
Yes
Enter glass id: id 55 match = 1
Yes
Enter glass id: id 56 match = 1
Yes
Enter glass id: id 57 match = 1
Yes
Enter glass id: id 58 match = 1
Yes
Enter glass id: id 59 match = 1
Yes
Enter glass id: id 60 match = 1
Yes
Enter glass id: id 61 match = 1
Yes
Enter glass id: id 62 match = 0
Yes
Enter glass id: id 63 match = 1
Yes
Enter glass id: id 64 match = 1
Yes
Enter glass id: id 65 match = 1
Yes
Enter glass id: id 66 match = 1
Yes
Enter glass id: id 67 match = 1
Yes
Enter glass id: id 68 match = 1
Yes
Enter glass id: id 69 match = 1
Yes
Enter glass id: id 70 match = 1
Yes
Enter glass id: id 71 match = 0
Yes
Enter glass id: id 72 match = 1
```

```
Yes
Enter glass id: id 73 match = 0
Yes
Enter glass id: id 74 match = 0
Yes
Enter glass id: id 75 match = 0
Yes
Enter glass id: id 76 match = 0
Yes
Enter glass id: id 77 match = 0
Yes
Enter glass id: id 78 match = 0
Yes
Enter glass id: id 79 match = 1
Yes
Enter glass id: id 80 match = 0
Yes
Enter glass id: id 81 match = 0
Yes
Enter glass id: id 82 match = 0
Yes
Enter glass id: id 83 match = 1
Yes
Enter glass id: id 84 match = 0
Yes
Enter glass id: id 85 match = 0
Yes
Enter glass id: id 86 match = 0
Yes
Enter glass id: id 87 match = 0
Yes
Enter glass id: id 88 match = 0
Yes
Enter glass id: id 89 match = 0
Yes
Enter glass id: id 90 match = 0
Yes
Enter glass id: id 91 match = 1
Yes
Enter glass id: id 92 match = 0
Yes
Enter glass id: id 93 match = 0
Yes
Enter glass id: id 94 match = 0
Yes
Enter glass id: id 95 match = 0
```

```
Yes
Enter glass id: id 96 match = 0
Yes
Enter glass id: id 97 match = 1
Yes
Enter glass id: id 98 match = 1
Yes
Enter glass id: id 99 match = 0
Yes
Enter glass id: id 100 match = 0
Yes
Enter glass id: id 101 match = 0
Yes
Enter glass id: id 102 match = 0
Yes
Enter glass id: id 103 match = 1
Yes
Enter glass id: id 104 match = 1
Yes
Enter glass id: id 105 match = 1
Yes
Enter glass id: id 106 match = 1
Yes
Enter glass id: id 107 match = 1
Yes
Enter glass id: id 108 match = 1
Yes
Enter glass id: id 109 match = 1
Yes
Enter glass id: id 110 match = 1
Yes
Enter glass id: id 111 match = 1
Yes
Enter glass id: id 112 match = 1
Yes
Enter glass id: id 113 match = 1
Yes
Enter glass id: id 114 match = 1
Yes
Enter glass id: id 115 match = 1
Yes
Enter glass id: id 116 match = 1
Yes
Enter glass id: id 117 match = 1
Yes
Enter glass id: id 118 match = 0
```

```
Yes
Enter glass id: id 119 match = 0
Yes
Enter glass id: id 120 match = 0
Yes
Enter glass id: id 121 match = 1
Yes
Enter glass id: id 122 match = 0
Yes
Enter glass id: id 123 match = 0
Yes
Enter glass id: id 124 match = 0
Yes
Enter glass id: id 125 match = 1
Yes
Enter glass id: id 126 match = 0
Yes
Enter glass id: id 127 match = 1
Yes
Enter glass id: id 128 match = 1
Yes
Enter glass id: id 129 match = 1
Yes
Enter glass id: id 130 match = 1
Yes
Enter glass id: id 131 match = 1
Yes
Enter glass id: id 132 match = 1
Yes
Enter glass id: id 133 match = 1
Yes
Enter glass id: id 134 match = 0
Yes
Enter glass id: id 135 match = 1
Yes
Enter glass id: id 136 match = 1
Yes
Enter glass id: id 137 match = 1
Yes
Enter glass id: id 138 match = 0
Yes
Enter glass id: id 139 match = 0
Yes
Enter glass id: id 140 match = 0
Yes
Enter glass id: id 141 match = 0
```

```
Yes
Enter glass id: id 142 match = 1
Yes
Enter glass id: id 143 match = 0
Yes
Enter glass id: id 144 match = 0
Yes
Enter glass id: id 145 match = 1
Yes
Enter glass id: id 146 match = 0
Yes
Enter glass id: id 147 match = 1
Yes
Enter glass id: id 148 match = 1
Yes
Enter glass id: id 149 match = 1
Yes
Enter glass id: id 150 match = 1
Yes
Enter glass id: id 151 match = 0
Yes
Enter glass id: id 152 match = 0
Yes
Enter glass id: id 153 match = 1
Yes
Enter glass id: id 154 match = 1
Yes
Enter glass id: id 155 match = 1
Yes
Enter glass id: id 156 match = 1
Yes
Enter glass id: id 157 match = 1
Yes
Enter glass id: id 158 match = 0
Yes
Enter glass id: id 159 match = 0
Yes
Enter glass id: id 160 match = 0
Yes
Enter glass id: id 161 match = 0
Yes
Enter glass id: id 162 match = 1
Yes
Enter glass id: id 163 match = 0
Yes
Enter glass id: id 164 match = 0
```

```
Yes
Enter glass id: id 165 match = 1
Yes
Enter glass id: id 166 match = 1
Yes
Enter glass id: id 167 match = 1
Yes
Enter glass id: id 168 match = 1
Yes
Enter glass id: id 169 match = 1
Yes
Enter glass id: id 170 match = 1
Yes
Enter glass id: id 171 match = 1
Yes
Enter glass id: id 172 match = 1
Yes
Enter glass id: id 173 match = 1
Yes
Enter glass id: id 174 match = 1
Yes
Enter glass id: id 175 match = 1
Yes
Enter glass id: id 176 match = 1
Yes
Enter glass id: id 177 match = 1
Yes
Enter glass id: id 178 match = 1
Yes
Enter glass id: id 179 match = 1
Yes
Enter glass id: id 180 match = 1
Yes
Enter glass id: id 181 match = 1
Yes
Enter glass id: id 182 match = 1
Yes
Enter glass id: id 183 match = 1
Yes
Enter glass id: id 184 match = 1
Yes
Enter glass id: id 185 match = 1
Yes
Enter glass id: id 186 match = 1
Yes
Enter glass id: id 187 match = 1
```

```
Yes
Enter glass id: id 188 match = 0
Yes
Enter glass id: id 189 match = 0
Yes
Enter glass id: id 190 match = 1
Yes
Enter glass id: id 191 match = 1
Yes
Enter glass id: id 192 match = 1
Yes
Enter glass id: id 193 match = 1
Yes
Enter glass id: id 194 match = 1
Yes
Enter glass id: id 195 match = 1
Yes
Enter glass id: id 196 match = 1
Yes
Enter glass id: id 197 match = 1
Yes
Enter glass id: id 198 match = 1
Yes
Enter glass id: id 199 match = 1
Yes
Enter glass id: id 200 match = 1
Yes
Enter glass id: id 201 match = 1
Yes
Enter glass id: id 202 match = 0
Yes
Enter glass id: id 203 match = 1
Yes
Enter glass id: id 204 match = 1
Yes
Enter glass id: id 205 match = 1
Yes
Enter glass id: id 206 match = 1
Yes
Enter glass id: id 207 match = 1
Yes
Enter glass id: id 208 match = 1
Yes
Enter glass id: id 209 match = 1
Yes
Enter glass id: id 210 match = 1
```

```
Yes
Enter glass id: id 211 match = 1
Yes
Enter glass id: id 212 match = 1
Yes
Enter glass id: id 213 match = 1
Yes
Enter glass id: id 214 match = 1
Yes
216 ?-
% halt
```

B.6 marry.pl output

```
Auomate file created ok.
About to run NN
NN run completed.
The neural network answer is correct for glassid 1, match = 1.
The neural network answer is correct for glassid 2, match = 1.
The neural network answer is wrong.
Will now try using prolog decision tree.

c:\everything\andrew\ou\tm426>c:\everything\andrew\ou\tm426\swiProlog\pl\bin\plcon  0<g5
Welcome to SWI-Prolog (Multi-threaded, Version 5.2.10)
Copyright (c) 1990-2003 University of Amsterdam.
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to redistribute it under certain conditions.
Please visit http://www.swi-prolog.org for details.

For help, use ?- help(Topic). or ?- apropos(Word).

1 ?- Warning: (c:/everything/andrew/ou/tm426/glass08.pl:1):
        Singleton variables: [Cat, Match]
Warning: (c:/everything/andrew/ou/tm426/glass08.pl:269):
        Singleton variables: [RefractiveIndex, Sodium, Magnesium, Aluminium, Potassium,
Warning: (c:/everything/andrew/ou/tm426/glass08.pl:275):
        Singleton variables: [RefractiveIndex, Sodium, Magnesium, Aluminium, Potassium,
Warning: (c:/everything/andrew/ou/tm426/glass08.pl:309):
        Singleton variables: [RefractiveIndex, Magnesium, Aluminium, Silicon, Potassium,
Warning: (c:/everything/andrew/ou/tm426/glass08.pl:315):
        Singleton variables: [RefractiveIndex, Magnesium, Aluminium, Silicon, Potassium,
Warning: (c:/everything/andrew/ou/tm426/glass08.pl:328):
        Singleton variables: [RefractiveIndex, Magnesium, Aluminium, Silicon, Potassium,
Warning: (c:/everything/andrew/ou/tm426/glass08.pl:334):
        Singleton variables: [Sodium, Magnesium, Aluminium, Silicon, Potassium, Calcium,
```

```
Warning: (c:/everything/andrew/ou/tm426/glass08.pl:340):
        Singleton variables: [Sodium, Magnesium, Aluminium, Silicon, Potassium, Calcium,
Warning: (c:/everything/andrew/ou/tm426/glass08.pl:374):
        Singleton variables: [Sodium, Magnesium, Aluminium, Silicon, Potassium, Calcium,
Warning: (c:/everything/andrew/ou/tm426/glass08.pl:394):
        Singleton variables: [RefractiveIndex, Sodium, Magnesium, Aluminium, Silicon, Po
Warning: (c:/everything/andrew/ou/tm426/glass08.pl:408):
        Singleton variables: [RefractiveIndex, Sodium, Magnesium, Silicon, Potassium, Ca
Warning: (c:/everything/andrew/ou/tm426/glass08.pl:414):
        Singleton variables: [RefractiveIndex, Sodium, Aluminium, Silicon, Potassium, Ca
Warning: (c:/everything/andrew/ou/tm426/glass08.pl:420):
        Singleton variables: [RefractiveIndex, Sodium, Aluminium, Silicon, Potassium, Ca
Warning: (c:/everything/andrew/ou/tm426/glass08.pl:440):
        Singleton variables: [RefractiveIndex, Sodium, Aluminium, Silicon, Potassium, Ca
Warning: (c:/everything/andrew/ou/tm426/glass08.pl:453):
        Singleton variables: [RefractiveIndex, Sodium, Magnesium, Aluminium, Potassium,
Warning: (c:/everything/andrew/ou/tm426/glass08.pl:459):
        Singleton variables: [RefractiveIndex, Sodium, Magnesium, Aluminium, Potassium,
Warning: (c:/everything/andrew/ou/tm426/glass08.pl:472):
        Singleton variables: [RefractiveIndex, Sodium, Aluminium, Silicon, Potassium, Ca
Warning: (c:/everything/andrew/ou/tm426/glass08.pl:492):
        Singleton variables: [Sodium, Magnesium, Aluminium, Silicon, Potassium, Calcium,
Warning: (c:/everything/andrew/ou/tm426/glass08.pl:498):
        Singleton variables: [Sodium, Magnesium, Aluminium, Silicon, Potassium, Calcium,
Warning: (c:/everything/andrew/ou/tm426/glass08.pl:504):
        Singleton variables: [RefractiveIndex, Sodium, Magnesium, Silicon, Potassium, Ca
Warning: (c:/everything/andrew/ou/tm426/glass08.pl:510):
        Singleton variables: [RefractiveIndex, Sodium, Magnesium, Silicon, Potassium, Ca
Warning: (c:/everything/andrew/ou/tm426/glass08.pl:516):
        Clauses of test2Si/10 are not together in the source-file
Warning: (c:/everything/andrew/ou/tm426/glass08.pl:523):
        Singleton variables: [RefractiveIndex, Sodium, Magnesium, Aluminium, Potassium,
Warning: (c:/everything/andrew/ou/tm426/glass08.pl:529):
        Singleton variables: [RefractiveIndex, Sodium, Magnesium, Aluminium, Silicon, Po
Warning: (c:/everything/andrew/ou/tm426/glass08.pl:535):
        Singleton variables: [RefractiveIndex, Sodium, Magnesium, Aluminium, Silicon, Po
Warning: (c:/everything/andrew/ou/tm426/glass08.pl:546):
        Singleton variables: [Cat, Category]
% glass08.pl compiled 0.01 sec, 48,124 bytes

Yes
id 3 match = 0

Yes
3 ?-
```

```
% halt
Prolog run finished.
The neural network answer is correct for glassid 4, match = 1.
The neural network answer is correct for glassid 5, match = 1.
The neural network answer is wrong.
Will now try using prolog decision tree.
```

Appendix C

Program listings

This section lists various files and program source.

C.1 readstring

The `read_string` predicate below was found to work on MS Windows and Linux and with Flex, SWI-prolog and GNU Prolog. It is thus platform and dialect neutral.

```
read_string(L):-
 get0(C), get_rest(C,L1), name(L,L1).
get_rest(10,[]):-
 !.
get_rest(13,[]):-
 !.
get_rest(C,[C|OUT]):-
  get0(C1), get_rest(C1,OUT).
```

C.2 glassnames

```
1,2,3,4,5,6,7.
id:ignore.
RI:continuous.
Na:continuous.
Mg:continuous.
Al:continuous.
Si:continuous.
K:continuous.
Ca:continuous.
Ba:continuous.
Fe:continuous.
```

C.3 mknni

```
#! /usr/bin/perl

# we need to parse the command line so we can get the
# number of attributes (9 or 16).

use strict;

sub outstring
{
    my $class_type;
    my $output;
    $class_type = $_[0];

    $output = "0 0 0 0 0 0 0";
    $output = "1 0 0 0 0 0 0" if ($class_type == 1);
    $output = "0 1 0 0 0 0 0" if ($class_type == 2);
    $output = "0 0 1 0 0 0 0" if ($class_type == 3);
    $output = "0 0 0 1 0 0 0" if ($class_type == 4);
    $output = "0 0 0 0 1 0 0" if ($class_type == 5);
    $output = "0 0 0 0 0 1 0" if ($class_type == 6);
    $output = "0 0 0 0 0 0 1" if ($class_type == 7);
    return $output;
};

sub initialise_min
{
    my @min_val;

    @min_val = (1.51115, 10.73, 0,    0.29, 69.81, 0, 5.43, 0, 0);

    return \@min_val;
};

sub initialise_max
{
    my @max_val;

# RI range is 1.51115 to 1.53393, not 1.5112 to 1.5339
  as stated in glass.names

    @max_val = (1.53393, 17.38, 4.49, 3.5,
                75.41, 6.21, 16.19, 3.15, 0.51);
    return \@max_val;
};
```

```
44:
45: sub main
46: {
47:     my $filename;
48:     my $line;
49:     my $min_ref;
50:     my $max_ref;
51:     my @attrs;
52:     my $i;
53:     my $count;
54:     my $ctype;
55:     my $normalise;
56:     my $n_attrs = 9;
57:
58:     $i = 0;
59:     if ($ARGV[0] eq "-z")
60:     {
61:         $i = 1;
62:         $normalise = 0;
63:     }
64:     else
65:     {
66:         $min_ref = initialise_min();
67:         $max_ref = initialise_max();
68:         $normalise = 1;
69:     }
70:
71:     $filename = $ARGV[$i];
72:     if ($filename eq "")
73:     {
74:         print "Error: Missing filename.\n";
75:         exit(1);
76:     }
77:
78:     open(FD, $filename) ||
           die("Error: Failed to open $filename for reading");
79:
80:     $count = 0;
81:     while ($line = <FD>)
82:     {
83:         chomp($line);
84:         $count = $count + 1;
85:         @attrs = split(/,/, $line);
86:         for ($i = 1; $i <= $n_attrs; $i = $i + 1)
87:         {
88:             if ($normalise == 1)
```

```
 89:             {
 90:                 if ($attrs[$i] < $min_ref->[$i-1] ||
 91:                     $attrs[$i] > $max_ref->[$i-1])
 92:                 {
 93:                     print "Error at line $count: ".
                            $attrs[$i]." is out of range (".
 94:                            $min_ref->[$i-1]." - ".
                                $max_ref->[$i-1].")\n";
 95:                     exit(1);
 96:                 }
 97:                 $attrs[$i] = ($attrs[$i] - $min_ref->[$i-1]) /
 98:                                     ($max_ref->[$i-1] - $min_ref->[$i-1]);
 99:             }
100:
101:             print $attrs[$i]." ";
102:         }
103:         $ctype = outstring($attrs[$n_attrs+1]);
104:         print $ctype."\n";
105:     }
106:
107:     close(FD);
108:     exit(0);
109: }
110:
111: main;
```

C.4 winner

```
 1: #! /usr/bin/perl
 2:
 3: use strict;
 4: sub main
 5: {
 6:     my $filename;
 7:     my $line;
 8:     my $rec_count;
 9:     my $match_count;
10:     my $outputs;
11:     my @values;
12:     my $onendx;
13:     my $maxndx;
14:     my $maxval;
15:     my $i;
16:     my $out2;
17:     my $score;
```

```
18:
19:     $filename = $ARGV[0];
20:     if ($filename eq "")
21:     {
22:         print "Error: Missing filename.\n";
23:         exit(1);
24:     }
25:
26:     open(FD, $filename) ||
              die("Unable to open $filename for reading");
27:
28:     $outputs = 7;
29:     $match_count = 0;
30:     $rec_count = 0;
31:     $out2 = $outputs * 2;
32:     while ($line = <FD>)
33:     {
34:         chomp($line);
35:         $rec_count = $rec_count + 1;
36:         @values = split(/\s+/, $line);
37:         $onendx = 0;
38:         for ($i = 0; $i < $outputs; $i = $i + 1)
39:         {
40:             if ($values[$i] == 1)
41:             {
42:                 $onendx = $i;
43:             }
44:         }
45:         $maxndx = $outputs;
46:         $maxval = $values[$outputs];
47:         for ($i = $outputs; $i < $out2; $i = $i + 1)
48:         {
49:             if ($values[$i] >= $values[$maxndx])
50:             {
51:                 $maxndx = $i;
52:                 $maxval = $values[$i];
53:             }
54:         }
55:         $maxndx = $maxndx - $outputs;
56:         if ($onendx == $maxndx)
57:         {
58:             $match_count = $match_count + 1;
59:         }
60:     }
61:
62:     close(FD);
```

```
63:
64:     $score = 100 * ($match_count/(0.0 + $rec_count));
65:
66:     print "There were $match_count matches out of ";
67:     print "$rec_count, $score percent.\n";
68:
69:     exit(0);
70: };
71:
72: main;
```

C.5 mkzoo

```
 1: #! /usr/bin/perl
 2:
 3: use strict;
 4:
 5: sub outstring
 6: {
 7:     my $class_type;
 8:     my $output;
 9:     $class_type = $_[0];
10:
11:     $output = "0 0 0 0 0 0 0";
12:     $output = "1 0 0 0 0 0 0" if ($class_type == 1);
13:     $output = "0 1 0 0 0 0 0" if ($class_type == 2);
14:     $output = "0 0 1 0 0 0 0" if ($class_type == 3);
15:     $output = "0 0 0 1 0 0 0" if ($class_type == 4);
16:     $output = "0 0 0 0 1 0 0" if ($class_type == 5);
17:     $output = "0 0 0 0 0 1 0" if ($class_type == 6);
18:     $output = "0 0 0 0 0 0 1" if ($class_type == 7);
19:     return $output;
20: };
21:
22: sub main
23: {
24:     my $filename;
25:     my $line;
26:     my @attrs;
27:     my $i;
28:     my $ctype;
29:     my $n_attrs = 16;
30:
31:     $filename = $ARGV[0];
32:     if ($filename eq "")
```

```
33:     {
34:         print "Error: Missing filename.\n";
35:         exit(1);
36:     }
37:
38:     open(FD, $filename) ||
             die("Error: Failed to open $filename for reading");
39:
40:     while ($line = <FD>)
41:     {
42:         chomp($line);
43:         @attrs = split(/,/, $line);
44:         for ($i = 1; $i <= $n_attrs; $i = $i + 1)
45:         {
46:             print $attrs[$i]." ";
47:         }
48:         $ctype = outstring($attrs[$n_attrs+1]);
49:         print $ctype."\n";
50:     }
51:
52:     close(FD);
53:     exit(0);
54: }
55:
56: main;
```

C.6 zoogen.pl

```
 1: #! /usr/bin/perl
 2:
 3: use strict;
 4:
 5: sub main
 6: {
 7:     my $line;
 8:     my $animal;
 9:     my $hair;
10:     my $feathers;
11:     my $eggs;
12:     my $milk;
13:     my $airborne;
14:     my $aquatic;
15:     my $predator;
16:     my $toothed;
17:     my $backbone;
```

```
18:     my $breathes;
19:     my $venomous;
20:     my $fins;
21:     my $legs;
22:     my $tail;
23:     my $domestic;
24:     my $catsize;
25:     my $category;
26:
27:     open (OUT_FD, ">zoo00.pro") || die ("Error: Cannot open "
28: ."output prolog source file");
29:
30:     print OUT_FD "% Prolog zoo classification program.\n";
31:     print OUT_FD "%\n";
32:     print OUT_FD "% Here are the animal attributes:\n";
33:     print OUT_FD "%\n";
34:
35:     open (FD, "zoo.data") || die ("Cannot open zoo.data for reading.");
36:     while($line = <FD>)
37:     {
38:         chomp($line);
39:         ($animal,$hair,$feathers,$eggs,$milk,$airborne,
40:          $aquatic,$predator,$toothed,$backbone,
41:          $breathes,$venomous,
42:          $fins,$legs,$tail,$domestic,$catsize,
43:          $category) = split(/,/, $line);
44:
45:         printf(OUT_FD "animal(%s,%s,%s,%s,%s,%s,%s,%s,%s,".
46:                       "%s,%s,%s,%s,%s,%s,%s,%s).\n",
47:          $animal,$hair,$feathers,$eggs,$milk,$airborne,
48:          $aquatic,$predator,$toothed,$backbone,
49:          $breathes,$venomous,
50:          $fins,$legs,$tail,$domestic,$catsize);
51:     }
52:     close(FD);
53:
54:     print OUT_FD "%\n";
55:     print OUT_FD "% Here are the classification rules\n";
56:     print OUT_FD "%\n";
57:
58:     close(OUT_FD);
59: }
60:
61: main;
```

C.7 glassgen.pl

```
 1: #! /usr/bin/perl
 2:
 3: use strict;
 4:
 5: sub main
 6: {
 7:     my $line;
 8:     my $category;
 9:     my $glassid;
10:     my $refractiveIndex;
11:     my $sodium;
12:     my $magnesium;
13:     my $aluminium;
14:     my $silicon;
15:     my $potassium;
16:     my $calcium;
17:     my $barium;
18:     my $iron;
19:
20:     open (OUT_FD, ">glass.pro") || die ("Error: Cannot open "
21: ."output prolog source file");
22:
23:     print OUT_FD "% Prolog glass classification program.\n";
24:     print OUT_FD "%\n";
25:     print OUT_FD "% Here are the glass attributes:\n";
26:     print OUT_FD "%\n";
27:
28:     open (FD, "glass.data") || die ("Cannot open glass.data for reading.");
29:     while($line = <FD>)
30:     {
31:         chomp($line);
32:         ($glassid
33:         ,$refractiveIndex,$sodium,$magnesium,$aluminium
34:         ,$silicon,$potassium,$calcium,
             $barium, $iron) = split(/,/, $line);
35:
36:         printf(OUT_FD "glass(%s,%s,%s,%s,%s,%s,%s,%s,%s,%s).\n",
37:                $glassid
38:               ,$refractiveIndex,$sodium,$magnesium,$aluminium
39:               ,$silicon,$potassium,$calcium,$barium,$iron);
40:     }
41:     close(FD);
42:
43:     print OUT_FD "\n%\n";
```

```
44:     print OUT_FD "% Here are the classification rules\n";
45:     print OUT_FD "%\n";
46:
47:     close(OUT_FD);
48: }
49:
50: main;
```

C.8 Exploring categories for zoo animals

C.8.1 cat4.pl

```
 1: #! /usr/bin/perl
 2:
 3: use strict;
 4:
 5: sub main
 6: {
 7:     my $line;
 8: my $animal;
 9: my $hair;
10: my $feathers;
11: my $eggs;
12: my $milk;
13: my $airborne;
14: my $aquatic;
15: my $predator;
16: my $toothed;
17: my $backbone;
18: my $breathes;
19: my $venomous;
20: my $fins;
21: my $legs;
22: my $tail;
23: my $domestic;
24: my $catsize;
25: my $category;
26:
27:     open (FD, "zoo.data") || die ("Cannot open zoo.data for reading.");
28:     while($line = <FD>)
29:     {
30:         chomp($line);
31:         ($animal,$hair,$feathers,$eggs,$milk,$airborne,
32:          $aquatic,$predator,$toothed,$backbone,$breathes,$venomous,
33:          $fins,$legs,$tail,$domestic,$catsize,
```

```
34:          $category) = split(/,/, $line);
35:         if ($hair == 0 && $feathers == 0 && $eggs == 1 &&
36:             $milk == 0 && $airborne == 0 &&
37:             $aquatic == 1 && $toothed == 1 &&
38:             $backbone == 1 && $breathes == 0 &&
39:             $fins == 1 && $legs == 0 && $tail == 1)
40:         {
41:             print $line."\n";
42:         }
43:     }
44:     close(FD);
45: }
46:
47: main;
```

C.8.2 cat5.pl

```
 1: #! /usr/bin/perl
 2:
 3: use strict;
 4:
 5: sub main
 6: {
 7:     my $line;
 8: my $animal;
 9: my $hair;
10: my $feathers;
11: my $eggs;
12: my $milk;
13: my $airborne;
14: my $aquatic;
15: my $predator;
16: my $toothed;
17: my $backbone;
18: my $breathes;
19: my $venomous;
20: my $fins;
21: my $legs;
22: my $tail;
23: my $domestic;
24: my $catsize;
25: my $category;
26:
27:     open (FD, "zoo.data") || die ("Cannot open zoo.data for reading.");
28:     while($line = <FD>)
29:     {
```

```
30:         chomp($line);
31:         ($animal,$hair,$feathers,$eggs,$milk,$airborne,
32:          $aquatic,$predator,$toothed,$backbone,$breathes,$venomous,
33:          $fins,$legs,$tail,$domestic,$catsize,
34:          $category) = split(/,/, $line);
35:         if ($hair == 0 && $feathers == 0 && $eggs == 1 &&
36:             $milk == 0 && $airborne == 0 &&
37:             $aquatic == 1 && $toothed == 1 && $backbone ==1 &&
38:             $breathes == 1 &&
39:             $fins == 0 && $legs == 4)
40:         {
41:             print $line."\n";
42:         }
43:     }
44:     close(FD);
45: }
46:
47: main;
```

C.8.3 cat6.pl

```
 1: #! /usr/bin/perl
 2:
 3: use strict;
 4:
 5: sub main
 6: {
 7:     my $line;
 8: my $animal;
 9: my $hair;
10: my $feathers;
11: my $eggs;
12: my $milk;
13: my $airborne;
14: my $aquatic;
15: my $predator;
16: my $toothed;
17: my $backbone;
18: my $breathes;
19: my $venomous;
20: my $fins;
21: my $legs;
22: my $tail;
23: my $domestic;
24: my $catsize;
```

```
25: my $category;
26:
27:     open (FD, "zoo.data") || die ("Cannot open zoo.data for reading.");
28:     while($line = <FD>)
29:     {
30:         chomp($line);
31:         ($animal,$hair,$feathers,$eggs,$milk,$airborne,
32:          $aquatic,$predator,$toothed,$backbone,$breathes,$venomous,
33:          $fins,$legs,$tail,$domestic,$catsize,
34:          $category) = split(/,/, $line);
35:         if ($feathers == 0 && $eggs == 1 &&
36:             $milk == 0 && $aquatic == 0 &&
37:             $toothed == 0 && $backbone == 0 &&
38:             $breathes == 1 &&
39:             $fins == 0 && $legs == 6)
40:         {
41:             print $line."\n";
42:         }
43:     }
44:     close(FD);
45: }
46:
47: main;
```

C.8.4 cat7.pl

```
 1: #! /usr/bin/perl
 2:
 3: use strict;
 4:
 5: sub main
 6: {
 7:     my $line;
 8: my $animal;
 9: my $hair;
10: my $feathers;
11: my $eggs;
12: my $milk;
13: my $airborne;
14: my $aquatic;
15: my $predator;
16: my $toothed;
17: my $backbone;
18: my $breathes;
19: my $venomous;
```

```
20: my $fins;
21: my $legs;
22: my $tail;
23: my $domestic;
24: my $catsize;
25: my $category;
26:
27:     open (FD, "zoo.data") || die ("Cannot open zoo.data for reading.");
28:     while($line = <FD>)
29:     {
30:         chomp($line);
31:         ($animal,$hair,$feathers,$eggs,$milk,$airborne,
32:          $aquatic,$predator,$toothed,$backbone,$breathes,$venomous,
33:          $fins,$legs,$tail,$domestic,$catsize,
34:          $category) = split(/,/, $line);
35:         if ($hair == 0 && $feathers == 0 &&
36:             $milk == 0 && $airborne == 0 &&
37:             $toothed == 0 && $backbone == 0 &&
38:             $fins == 0 && $domestic == 0 &&
39:            ($predator == 1 || ($predator ==0 && $legs == 0)))
40:         {
41:             print $line."\n";
42:         }
43:     }
44:     close(FD);
45: }
46:
47: main;
```

C.8.5 zoo01.pro

```
 1: % Prolog zoo classification program.
 2: %
 3: % Here are the animal attributes:
 4: %
 5: animal(aardvark,1,0,0,1,0,0,1,1,1,1,0,0,4,0,0,1).
 6: animal(antelope,1,0,0,1,0,0,0,1,1,1,0,0,4,1,0,1).
 7: animal(bass,0,0,1,0,0,1,1,1,1,0,0,1,0,1,0,0).
 8: animal(bear,1,0,0,1,0,0,1,1,1,1,0,0,4,0,0,1).
 9: animal(boar,1,0,0,1,0,0,1,1,1,1,0,0,4,1,0,1).
10: animal(buffalo,1,0,0,1,0,0,0,1,1,1,0,0,4,1,0,1).
11: animal(calf,1,0,0,1,0,0,0,1,1,1,0,0,4,1,1,1).
12: animal(carp,0,0,1,0,0,1,0,1,1,0,0,1,0,1,1,0).
13: animal(catfish,0,0,1,0,0,1,1,1,1,0,0,1,0,1,0,0).
14: animal(cavy,1,0,0,1,0,0,0,1,1,1,0,0,4,0,1,0).
15: animal(cheetah,1,0,0,1,0,0,1,1,1,1,0,0,4,1,0,1).
```

```
16: animal(chicken,0,1,1,0,1,0,0,0,1,1,0,0,2,1,1,0).
17: animal(chub,0,0,1,0,0,1,1,1,1,0,0,1,0,1,0,0).
18: animal(clam,0,0,1,0,0,0,0,1,0,0,0,0,0,0,0,0,0).
19: animal(crab,0,0,1,0,0,0,1,1,0,0,0,0,0,0,4,0,0,0).
20: animal(crayfish,0,0,1,0,0,0,1,1,0,0,0,0,0,0,6,0,0,0).
21: animal(crow,0,1,1,0,1,0,1,0,1,1,0,0,2,1,0,0).
22: animal(deer,1,0,0,1,0,0,0,0,1,1,1,0,0,4,1,0,1).
23: animal(dogfish,0,0,1,0,0,0,1,1,1,1,0,0,1,0,1,0,1).
24: animal(dolphin,0,0,0,1,0,1,1,1,1,1,0,1,0,1,0,1).
25: animal(dove,0,1,1,0,1,0,0,0,0,1,1,0,0,2,1,1,0).
26: animal(duck,0,1,1,0,0,1,1,0,0,0,1,1,0,0,2,1,0,0).
27: animal(elephant,1,0,0,1,0,0,0,0,1,1,1,0,0,4,1,0,1).
28: animal(flamingo,0,1,1,0,1,0,0,0,0,1,1,0,0,2,1,0,1).
29: animal(flea,0,0,1,0,0,0,0,0,0,0,1,0,0,6,0,0,0).
30: animal(frog,0,0,1,0,0,0,1,1,1,1,1,0,0,4,0,0,0).
31: animal(frog,0,0,1,0,0,0,1,1,1,1,1,1,0,4,0,0,0).
32: animal(fruitbat,1,0,0,1,1,0,0,0,1,1,1,0,0,2,1,0,0).
33: animal(giraffe,1,0,0,1,0,0,0,0,1,1,1,0,0,4,1,0,1).
34: animal(girl,1,0,0,1,0,0,0,1,1,1,1,0,0,2,0,1,1).
35: animal(gnat,0,0,1,0,1,0,0,0,0,0,1,0,0,6,0,0,0).
36: animal(goat,1,0,0,1,0,0,0,0,1,1,1,0,0,4,1,1,1).
37: animal(gorilla,1,0,0,1,0,0,0,0,1,1,1,0,0,2,0,0,1).
38: animal(gull,0,1,1,0,1,1,1,0,0,1,1,0,0,2,1,0,0).
39: animal(haddock,0,0,1,0,0,1,0,1,1,0,0,1,0,1,0,0).
40: animal(hamster,1,0,0,1,0,0,0,0,1,1,1,0,0,4,1,1,0).
41: animal(hare,1,0,0,1,0,0,0,0,1,1,1,0,0,4,1,0,0).
42: animal(hawk,0,1,1,0,1,0,1,0,1,1,0,0,2,1,0,0).
43: animal(herring,0,0,1,0,0,1,1,1,1,0,0,1,0,1,0,0).
44: animal(honeybee,1,0,1,0,1,0,0,0,0,0,1,1,0,6,0,1,0).
45: animal(housefly,1,0,1,0,1,0,0,0,0,0,1,0,0,6,0,0,0).
46: animal(kiwi,0,1,1,0,0,0,1,0,1,1,0,0,2,1,0,0).
47: animal(ladybird,0,0,1,0,1,0,1,0,0,1,0,0,6,0,0,0).
48: animal(lark,0,1,1,0,1,0,0,0,1,1,0,0,2,1,0,0).
49: animal(leopard,1,0,0,1,0,0,1,1,1,1,0,0,4,1,0,1).
50: animal(lion,1,0,0,1,0,0,1,1,1,1,0,0,4,1,0,1).
51: animal(lobster,0,0,1,0,0,1,1,0,0,0,0,0,6,0,0,0).
52: animal(lynx,1,0,0,1,0,0,1,1,1,1,0,0,4,1,0,1).
53: animal(mink,1,0,0,1,0,1,1,1,1,1,0,0,4,1,0,1).
54: animal(mole,1,0,0,1,0,0,1,1,1,1,0,0,4,1,0,0).
55: animal(mongoose,1,0,0,1,0,0,1,1,1,1,0,0,4,1,0,1).
56: animal(moth,1,0,1,0,1,0,0,0,0,1,0,0,6,0,0,0).
57: animal(newt,0,0,1,0,0,1,1,1,1,1,0,0,4,1,0,0).
58: animal(octopus,0,0,1,0,0,1,1,0,0,0,0,0,8,0,0,1).
59: animal(opossum,1,0,0,1,0,0,1,1,1,1,0,0,4,1,0,0).
60: animal(oryx,1,0,0,1,0,0,0,0,1,1,1,0,0,4,1,0,1).
61: animal(ostrich,0,1,1,0,0,0,0,0,0,1,1,0,0,2,1,0,1).
```

```
 62: animal(parakeet,0,1,1,0,1,0,0,0,1,1,0,0,2,1,1,0).
 63: animal(penguin,0,1,1,0,0,1,1,0,1,1,0,0,2,1,0,1).
 64: animal(pheasant,0,1,1,0,1,0,0,0,1,1,0,0,2,1,0,0).
 65: animal(pike,0,0,1,0,0,1,1,1,1,0,0,1,0,1,0,1).
 66: animal(piranha,0,0,1,0,0,1,1,1,1,0,0,1,0,1,0,0).
 67: animal(pitviper,0,0,1,0,0,0,1,1,1,1,1,0,0,1,0,0).
 68: animal(platypus,1,0,1,1,0,1,1,0,1,1,0,0,4,1,0,1).
 69: animal(polecat,1,0,0,1,0,0,1,1,1,1,0,0,4,1,0,1).
 70: animal(pony,1,0,0,1,0,0,0,1,1,1,0,0,4,1,1,1).
 71: animal(porpoise,0,0,0,1,0,1,1,1,1,1,0,1,0,1,0,1).
 72: animal(puma,1,0,0,1,0,0,1,1,1,1,0,0,4,1,0,1).
 73: animal(pussycat,1,0,0,1,0,0,1,1,1,1,0,0,4,1,1,1).
 74: animal(raccoon,1,0,0,1,0,0,1,1,1,1,0,0,4,1,0,1).
 75: animal(reindeer,1,0,0,1,0,0,0,1,1,1,0,0,4,1,1,1).
 76: animal(rhea,0,1,1,0,0,0,1,0,1,1,0,0,2,1,0,1).
 77: animal(scorpion,0,0,0,0,0,0,1,0,0,1,1,0,8,1,0,0).
 78: animal(seahorse,0,0,1,0,0,1,0,1,1,0,0,1,0,1,0,0).
 79: animal(seal,1,0,0,1,0,1,1,1,1,1,0,1,0,0,0,1).
 80: animal(sealion,1,0,0,1,0,1,1,1,1,1,0,1,2,1,0,1).
 81: animal(seasnake,0,0,0,0,0,1,1,1,1,0,1,0,0,1,0,0).
 82: animal(seawasp,0,0,1,0,0,1,1,0,0,0,1,0,0,0,0,0).
 83: animal(skimmer,0,1,1,0,1,1,1,0,1,1,0,0,2,1,0,0).
 84: animal(skua,0,1,1,0,1,1,1,0,1,1,0,0,2,1,0,0).
 85: animal(slowworm,0,0,1,0,0,0,1,1,1,1,0,0,0,1,0,0).
 86: animal(slug,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0).
 87: animal(sole,0,0,1,0,0,1,0,1,1,0,0,1,0,1,0,0).
 88: animal(sparrow,0,1,1,0,1,0,0,0,1,1,0,0,2,1,0,0).
 89: animal(squirrel,1,0,0,1,0,0,0,1,1,1,0,0,2,1,0,0).
 90: animal(starfish,0,0,1,0,0,1,1,0,0,0,0,0,5,0,0,0).
 91: animal(stingray,0,0,1,0,0,1,1,1,1,0,1,1,0,1,0,1).
 92: animal(swan,0,1,1,0,1,1,0,0,1,1,0,0,2,1,0,1).
 93: animal(termite,0,0,1,0,0,0,0,0,0,1,0,0,6,0,0,0).
 94: animal(toad,0,0,1,0,0,1,0,1,1,1,0,0,4,0,0,0).
 95: animal(tortoise,0,0,1,0,0,0,0,0,0,1,1,0,0,4,1,0,1).
 96: animal(tuatara,0,0,1,0,0,0,1,1,1,1,0,0,4,1,0,0).
 97: animal(tuna,0,0,1,0,0,1,1,1,1,0,0,1,0,1,0,1).
 98: animal(vampire,1,0,0,1,1,0,0,1,1,1,0,0,2,1,0,0).
 99: animal(vole,1,0,0,1,0,0,0,1,1,1,0,0,4,1,0,0).
100: animal(vulture,0,1,1,0,1,0,1,0,1,1,0,0,2,1,0,1).
101: animal(wallaby,1,0,0,1,0,0,0,1,1,1,0,0,2,1,0,1).
102: animal(wasp,1,0,1,0,1,0,0,0,0,1,1,0,6,0,0,0).
103: animal(wolf,1,0,0,1,0,0,1,1,1,1,0,0,4,1,0,1).
104: animal(worm,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0).
105: animal(wren,0,1,1,0,1,0,0,0,1,1,0,0,2,1,0,0).
106:
107: %
```

```
% Here are the classification rules
%
go:-
    Cat = 0,
    identify(Animal).

identify(Animal):-
    write('Enter animal name: '),
    read_string(Animal),
    animal(Animal, Hair, Feathers, Eggs, Milk, Airborne,
                  Aquatic, Predator, Toothed, Backbone, Breathes,
                  Venomous, Fins, Legs, Tail, Domestic, Catsize),
    print_attributes(
                  Hair, Feathers, Eggs, Milk, Airborne,
                  Aquatic, Predator, Toothed, Backbone, Breathes,
                  Venomous, Fins, Legs, Tail, Domestic,
                  Catsize),
    find_category(Hair, Feathers, Eggs, Milk, Airborne,
                  Aquatic, Predator, Toothed, Backbone, Breathes,
                  Venomous, Fins, Legs, Tail, Domestic, Catsize,
                  Cat),
    write('Category is '), write(Cat), nl.

read_string(L):-
 get0(C), get_rest(C,L1), name(L,L1).
get_rest(10,[]):-
 !.
get_rest(13,[]):-
 !.
get_rest(C,[C|OUT]):-
  get0(C1), get_rest(C1,OUT).

print_attributes(Hair, Feathers, Eggs, Milk, Airborne,
                  Aquatic, Predator, Toothed, Backbone, Breathes,
                  Venomous, Fins, Legs, Tail, Domestic,
                  Catsize):-
nl,
write('hair      = '), write(Hair),     write('   '),
write('feathers  = '), write(Feathers), write('   '),
write('eggs      = '), write(Eggs),     write('   '),
write('milk      = '), write(Milk),     write('   '), nl,
write('airborne  = '), write(Airborne), write('   '),
write('aquatic   = '), write(Aquatic),  write('   '),
write('predator  = '), write(Predator), write('   '),
write('toothed   = '), write(Toothed),  write('   '), nl,
write('backbone  = '), write(Backbone), write('   '),
```

```
154: write('breathes  = '), write(Breathes), write('   '),
155: write('venomous  = '), write(Venomous), write('   '),
156: write('fins      = '), write(Fins),     write('   '), nl,
157: write('legs      = '), write(Legs),     write('   '),
158: write('tail      = '), write(Tail),     write('   '),
159: write('domestic  = '), write(Domestic), write('   '),
160: write('catsize   = '), write(Catsize),  write('   '), nl
161: .
162:
163: find_category(Hair, Feathers, Eggs,
164:               Milk, Airborne, Aquatic,
165:               Predator, Toothed,
166:               Backbone, Breathes, Venomous,
167:               Fins, Legs, Tail,
168:               Domestic, Catsize, Cat):-
169:     Feathers is 0, Milk is 1,
170:     Backbone is 1, Breathes is 1, Venomous is 0,
171:     cat1hairy(Hair,Aquatic,Fins),
172:     cat1toothed(Toothed, Eggs),
173:     Cat = 1
174: .
175:
176: find_category(Hair, Feathers, Eggs, Milk, Airborne, Aquatic,
177:               Predator, Toothed, Backbone, Breathes, Venomous,
178:               Fins, Legs, Tail, Domestic, Catsize, Cat):-
179: Feathers is 1, Eggs = 1, Milk is 0,
180: Toothed is 0, Backbone is 1, Breathes is 1, Venomous is 0,
181: Fins is 0,
182: Cat = 2
183: .
184:
185: find_category(Hair, Feathers, Eggs, Milk, Airborne, Aquatic,
186:               Predator, Toothed, Backbone, Breathes, Venomous,
187:               Fins, Legs, Tail, Domestic, Catsize, Cat):-
188: Hair is 0, Feathers is 0, Milk is 0, Airborne is 0,
189: Backbone is 1, Fins is 0, Tail is 1, Domestic is 0,
190: cat3eggs(Eggs,Aquatic),
191: Cat = 3
192: .
193:
194: find_category(Hair, Feathers, Eggs, Milk, Airborne, Aquatic,
195:               Predator, Toothed, Backbone, Breathes, Venomous,
196:               Fins, Legs, Tail, Domestic, Catsize, Cat):-
197: Hair is 0, Feathers is 0, Eggs is 1, Milk is 0,
198: Airborne is 0, Aquatic is 1, Toothed is 1, Backbone is 1,
199: Breathes is 0, Fins is 1, Legs is 0, Tails is 1,
```

```
200: Cat = 4
201: .
202:
203: find_category(Hair, Feathers, Eggs, Milk, Airborne, Aquatic,
204:               Predator, Toothed, Backbone, Breathes, Venomous,
205:               Fins, Legs, Tail, Domestic, Catsize, Cat):-
206: Hair is 0, Feathers is 0, Eggs is 1, Milk is 0,
207: Airborne is 0, Aquatic is 1, Toothed is 1, Backbone is 1,
208: Fins is 0, Legs is 4,
209: Cat = 5
210: .
211:
212: find_category(Hair, Feathers, Eggs, Milk, Airborne, Aquatic,
213:               Predator, Toothed, Backbone, Breathes, Venomous,
214:               Fins, Legs, Tail, Domestic, Catsize, Cat):-
215: Feathers is 0, Eggs is 1, Milk is 0, Aquatic is 0,
216: Toothed is 0, Backbone is 0, Breathes is 1,
217: Fins is 0, Legs is 6,
218: Cat = 6
219: .
220:
221: find_category(Hair, Feathers, Eggs, Milk, Airborne, Aquatic,
222:               Predator, Toothed, Backbone, Breathes, Venomous,
223:               Fins, Legs, Tail, Domestic, Catsize, Cat):-
224: Hair is 0, Feathers is 0, Milk is 0, Airborne is 0,
225: Toothed is 0, Backbone is 0, Fins is 0, Domestic is 0,
226: cat7predator(Predator, Legs),
227: Cat = 7
228: .
229:
230: cat1hairy(Hair,Aquatic,Fins):-
231: Hair is 1; Hair is 0, Aquatic is 1, Fins is 1.
232:
233: cat1toothed(Toothed, Eggs):-
234: Toothed is 1, Eggs is 0; Toothed is 0, Eggs is 1.
235:
236: cat3eggs(Eggs, Aquatic):-
237: Eggs is 1, Aquatic is 0
238: .
239:
240: cat7predator(Predator, Legs):-
241: Predator is 1; Predator is 0, Legs is 0
242: .
243:
```

C.8.6 zoo02.pro

```
106 %
107 % Here are the classification rules
108 %
109 go:-
110     Cat = 0,
111     Match = 0,
112     identify(Animal).
113
114 identify(Animal):-
115     write('Enter animal name: '),
116     read_string(Animal),
117     animal(Animal, Hair, Feathers, Eggs, Milk, Airborne,
118                   Aquatic, Predator, Toothed, Backbone, Breathes,
119                   Venomous, Fins, Legs, Tail, Domestic,
120                   Catsize, Category),
121     find_category(Hair, Feathers, Eggs, Milk, Airborne,
122                   Aquatic, Predator, Toothed, Backbone, Breathes,
123                   Venomous, Fins, Legs, Tail, Domestic, Catsize,
124                   Cat),
125     write(Animal), write(' category is '), write(Cat),
126     find_match(Cat, Category, Match),
127     write(' match = '), write(Match), nl
128 .
129
130 read_string(L):-
131  get0(C), get_rest(C,L1), name(L,L1).
132 get_rest(10,[]):-
133  !.
134 get_rest(13,[]):-
135  !.
136 get_rest(C,[C|OUT]):-
137   get0(C1), get_rest(C1,OUT).
138
139 find_category(Hair, Feathers, Eggs,
140               Milk, Airborne, Aquatic,
141               Predator, Toothed,
142               Backbone, Breathes, Venomous,
143               Fins, Legs, Tail,
144               Domestic, Catsize, Cat):-
145     Feathers is 0, Milk is 1,
146     Backbone is 1, Breathes is 1, Venomous is 0,
147     cat1hairy(Hair,Aquatic,Fins),
148     cat1toothed(Toothed, Eggs),
149     Cat = 1
```

```
150 .
151
152 find_category(Hair, Feathers, Eggs, Milk, Airborne, Aquatic,
153               Predator, Toothed, Backbone, Breathes, Venomous,
154               Fins, Legs, Tail, Domestic, Catsize, Cat):-
155 Feathers is 1, Eggs = 1, Milk is 0,
156 Toothed is 0, Backbone is 1, Breathes is 1, Venomous is 0,
157 Fins is 0,
158 Cat = 2
159 .
160
161 find_category(Hair, Feathers, Eggs, Milk, Airborne, Aquatic,
162               Predator, Toothed, Backbone, Breathes, Venomous,
163               Fins, Legs, Tail, Domestic, Catsize, Cat):-
164 Hair is 0, Feathers is 0, Milk is 0, Airborne is 0,
165 Backbone is 1, Fins is 0, Tail is 1, Domestic is 0,
166 cat3eggs(Eggs,Aquatic),
167 Cat = 3
168 .
169
170 find_category(Hair, Feathers, Eggs, Milk, Airborne, Aquatic,
171               Predator, Toothed, Backbone, Breathes, Venomous,
172               Fins, Legs, Tail, Domestic, Catsize, Cat):-
173 Hair is 0, Feathers is 0, Eggs is 1, Milk is 0,
174 Airborne is 0, Aquatic is 1, Toothed is 1, Backbone is 1,
175 Breathes is 0, Fins is 1, Legs is 0, Tails is 1,
176 Cat = 4
177 .
178
179 find_category(Hair, Feathers, Eggs, Milk, Airborne, Aquatic,
180               Predator, Toothed, Backbone, Breathes, Venomous,
181               Fins, Legs, Tail, Domestic, Catsize, Cat):-
182 Hair is 0, Feathers is 0, Eggs is 1, Milk is 0,
183 Airborne is 0, Aquatic is 1, Toothed is 1, Backbone is 1,
184 Fins is 0, Legs is 4,
185 Cat = 5
186 .
187
188 find_category(Hair, Feathers, Eggs, Milk, Airborne, Aquatic,
189               Predator, Toothed, Backbone, Breathes, Venomous,
190               Fins, Legs, Tail, Domestic, Catsize, Cat):-
191 Feathers is 0, Eggs is 1, Milk is 0, Aquatic is 0,
192 Toothed is 0, Backbone is 0, Breathes is 1,
193 Fins is 0, Legs is 6,
194 Cat = 6
195 .
```

```
196
197 find_category(Hair, Feathers, Eggs, Milk, Airborne, Aquatic,
198               Predator, Toothed, Backbone, Breathes, Venomous,
199               Fins, Legs, Tail, Domestic, Catsize, Cat):-
200 Hair is 0, Feathers is 0, Milk is 0, Airborne is 0,
201 Toothed is 0, Backbone is 0, Fins is 0, Domestic is 0,
202 cat7predator(Predator, Legs),
203 Cat = 7
204 .
205
206 cat1hairy(Hair,Aquatic,Fins):-
207 Hair is 1; Hair is 0, Aquatic is 1, Fins is 1.
208
209 cat1toothed(Toothed, Eggs):-
210 Toothed is 1, Eggs is 0; Toothed is 0, Eggs is 1.
211
212 cat3eggs(Eggs, Aquatic):-
213 Eggs is 1, Aquatic is 0
214 .
215
216 cat7predator(Predator, Legs):-
217 Predator is 1; Predator is 0, Legs is 0
218 .
219
220 find_match(Cat, Category, Match):-
221 Cat is Category,
222 Match = 1
223 .
224
225 find_match(Cat, Category, Match):-
226 Match is 0,
227 Match = 0
228 .
```

C.8.7 zoo02play.pl

```
 1 #! /usr/bin/perl
 2
 3 use strict;
 4
 5 sub main
 6 {
 7     my $line;
 8     my $animal;
 9     my $hair;
10     my $feathers;
```

```
11      my $eggs;
12      my $milk;
13      my $airborne;
14      my $aquatic;
15      my $predator;
16      my $toothed;
17      my $backbone;
18      my $breathes;
19      my $venomous;
20      my $fins;
21      my $legs;
22      my $tail;
23      my $domestic;
24      my $catsize;
25      my $category;
26
27      open (OUT_FD, ">zoo02play") || die ("Error: Cannot open output script file");
28
29      print OUT_FD "#! /bin/sh\n";
30      print OUT_FD "\n";
31      print OUT_FD "pl <<EOF\n";
32      print OUT_FD "['zoo02.pro'].\n";
33
34      open (FD, "zoo.data") || die ("Cannot open zoo.data for reading.");
35      while($line = <FD>)
36      {
37          chomp($line);
38          ($animal,$hair,$feathers,$eggs,$milk,$airborne,
39           $aquatic,$predator,$toothed,$backbone,$breathes,$venomous,
40           $fins,$legs,$tail,$domestic,$catsize,
41           $category) = split(/,/, $line);
42          print OUT_FD "go.\n";
43          printf(OUT_FD "%s\n", $animal);
44      }
45      close(FD);
46
47      print OUT_FD "EOF\n";
48      print OUT_FD "\n";
49
50      close(OUT_FD);
51  }
52
53  main;
```

C.8.8 zoo03.pro

Note: the facts section is omiited for brevity in the listing below, since it is identical to the one above.

```
107: %
108: % Here are the classification rules
109: %
110: go:-
111:     Cat = 0,
112:     identify(Animal).
113:
114: identify(Animal):-
115:     write('Enter animal name: '),
116:     read_string(Animal),
117:     animal(Animal, Hair, Feathers, Eggs, Milk, Airborne,
118:                    Aquatic, Predator, Toothed, Backbone, Breathes,
119:                    Venomous, Fins, Legs, Tail, Domestic, Catsize),
120:     print_attributes(
121:                    Hair, Feathers, Eggs, Milk, Airborne,
122:                    Aquatic, Predator, Toothed, Backbone, Breathes,
123:                    Venomous, Fins, Legs, Tail, Domestic,
124:                    Catsize),
125:     find_category(Hair, Feathers, Eggs, Milk, Airborne,
126:                    Aquatic, Predator, Toothed, Backbone, Breathes,
127:                    Venomous, Fins, Legs, Tail, Domestic, Catsize,
128:                    Cat),
129:     write('Category is '), write(Cat), nl.
130:
131: read_string(L):-
132:  get0(C), get_rest(C,L1), name(L,L1).
133: get_rest(10,[]):-
134:  !.
135: get_rest(13,[]):-
136:  !.
137: get_rest(C,[C|OUT]):-
138:   get0(C1), get_rest(C1,OUT).
139:
140: print_attributes(Hair, Feathers, Eggs, Milk, Airborne,
141:                    Aquatic, Predator, Toothed, Backbone, Breathes,
142:                    Venomous, Fins, Legs, Tail, Domestic,
143:                    Catsize):-
144: nl,
145: write('hair      = '), write(Hair),     write('   '),
146: write('feathers  = '), write(Feathers), write('   '),
147: write('eggs      = '), write(Eggs),     write('   '),
148: write('milk      = '), write(Milk),     write('   '), nl,
```

```
149: write('airborne  = '), write(Airborne), write('   '),
150: write('aquatic   = '), write(Aquatic),  write('   '),
151: write('predator  = '), write(Predator), write('   '),
152: write('toothed   = '), write(Toothed),  write('   '), nl,
153: write('backbone  = '), write(Backbone), write('   '),
154: write('breathes  = '), write(Breathes), write('   '),
155: write('venomous  = '), write(Venomous), write('   '),
156: write('fins      = '), write(Fins),     write('   '), nl,
157: write('legs      = '), write(Legs),     write('   '),
158: write('tail      = '), write(Tail),     write('   '),
159: write('domestic  = '), write(Domestic), write('   '),
160: write('catsize   = '), write(Catsize),  write('   '), nl
161: .
162:
163: find_category(Hair, Feathers, Eggs,
164:               Milk, Airborne, Aquatic,
165:               Predator, Toothed,
166:               Backbone, Breathes, Venomous,
167:               Fins, Legs, Tail,
168:               Domestic, Catsize, Cat):-
169:     Milk is 1,
170:     Cat = 1
171: .
172:
173: find_category(Hair, Feathers, Eggs, Milk, Airborne, Aquatic,
174:               Predator, Toothed, Backbone, Breathes, Venomous,
175:               Fins, Legs, Tail, Domestic, Catsize, Cat):-
176:     Fins is 1,
177:     Cat = 4
178: .
179:
180: find_category(Hair, Feathers, Eggs, Milk, Airborne, Aquatic,
181:               Predator, Toothed, Backbone, Breathes, Venomous,
182:               Fins, Legs, Tail, Domestic, Catsize, Cat):-
183:     Feathers is 1,
184:     Cat = 2
185: .
186:
187: find_category(Hair, Feathers, Eggs, Milk, Airborne, Aquatic,
188:               Predator, Toothed, Backbone, Breathes, Venomous,
189:               Fins, Legs, Tail, Domestic, Catsize, Cat):-
190:     Backbone is 1, Tails is 1,
191:     Cat = 3
192: .
193:
194: find_category(Hair, Feathers, Eggs, Milk, Airborne, Aquatic,
```

```
195:               Predator, Toothed, Backbone, Breathes, Venomous,
196:               Fins, Legs, Tail, Domestic, Catsize, Cat):-
197:     Backbone is 1, Tails is 0,
198:     Cat = 5
199: .
200:
201: find_category(Hair, Feathers, Eggs, Milk, Airborne, Aquatic,
202:               Predator, Toothed, Backbone, Breathes, Venomous,
203:               Fins, Legs, Tail, Domestic, Catsize, Cat):-
204:     Airborne is 1,
205:     Cat = 6
206: .
207:
208: find_category(Hair, Feathers, Eggs, Milk, Airborne, Aquatic,
209:               Predator, Toothed, Backbone, Breathes, Venomous,
210:               Fins, Legs, Tail, Domestic, Catsize, Cat):-
211:     Predator is 1,
212:     Cat = 7
213: .
214:
215: find_category(Hair, Feathers, Eggs, Milk, Airborne, Aquatic,
216:               Predator, Toothed, Backbone, Breathes, Venomous,
217:               Fins, Legs, Tail, Domestic, Catsize, Cat):-
218:     Legs is 0,
219:     Cat = 7
220: .
221:
222: find_category(Hair, Feathers, Eggs, Milk, Airborne, Aquatic,
223:               Predator, Toothed, Backbone, Breathes, Venomous,
224:               Fins, Legs, Tail, Domestic, Catsize, Cat):-
225:     Cat = 6
226: .
227:
```

C.9 Exploring categories with the glass dataset

The Prolog program listings omit the glass facts statements since they are very large, not easily verified, and obscure the importance of the other Prolog statements.

C.9.1 glass01.pro

```
220: %
221: % Here are the classification rules
222: %
```

```
223: go:-
224:     Cat = 0,
225:     identify(Gid).
226:
227: identify(Gid):-
228:     write('Enter glass id: '),
229:     read_string(Gid),
230:     glass(Gid,
231:           RefractiveIndex, Sodium, Magnesium, Aluminium,
232:           Silicon, Potassium, Calcium, Barium, Iron
233:           ),
234:     print_attributes(
235:           RefractiveIndex, Sodium, Magnesium, Aluminium,
236:           Silicon, Potassium, Calcium, Barium, Iron
237:                     ),
238:     find_category(
239:           RefractiveIndex, Sodium, Magnesium, Aluminium,
240:           Silicon, Potassium, Calcium, Barium, Iron,
241:                   Cat),
242:     write('Category is '), write(Cat), nl.
243:
244: read_string(L):-
245:  get0(C), get_rest(C,L1), name(L,L1).
246: get_rest(10,[]):-
247:  !.
248: get_rest(13,[]):-
249:  !.
250: get_rest(C,[C|OUT]):-
251:   get0(C1), get_rest(C1,OUT).
252:
253: print_attributes(
254:           RefractiveIndex, Sodium, Magnesium, Aluminium,
255:           Silicon, Potassium, Calcium, Barium, Iron
256:                 ):-
257: nl,
258: write('Refractive index = '), write(RefractiveIndex), write('   '),nl,
259: write('Sodium           = '), write(Sodium), write('   '),nl,
260: write('Magnesium        = '), write(Magnesium), write('   '),nl,
261: write('Aluminium        = '), write(Aluminium),write('   '),nl,
262: write('Silicon          = '), write(Silicon), write('   '), nl,
263: write('Potassium        = '), write(Potassium), write('   '),nl,
264: write('Calcium          = '), write(Calcium), write('   '),nl,
265: write('Barium           = '), write(Barium), write('   '),nl,
266: write('Iron             = '), write(Iron), write('   '), nl
267: .
268:
```

```
269: find_category(
270:           RefractiveIndex, Sodium, Magnesium, Aluminium,
271:           Silicon, Potassium, Calcium, Barium, Iron
272:         , Cat
273:             ):-
274:     RefractiveIndex > 0.007, RefractiveIndex < 0.527, Iron < 0.1765,
275:     Cat = 7
276: .
```

C.9.2 glass02.pro

```
220: %
221: % Here are the classification rules
222: %
223: go:-
224:     Cat = 0, Node = 0,
225:     identify(Gid).
226:
227: identify(Gid):-
228:     write('Enter glass id: '),
229:     read_string(Gid),
230:     glass(Gid,
231:           RefractiveIndex, Sodium, Magnesium, Aluminium,
232:           Silicon, Potassium, Calcium, Barium, Iron
233:           ),
234:     print_attributes(
235:           RefractiveIndex, Sodium, Magnesium, Aluminium,
236:           Silicon, Potassium, Calcium, Barium, Iron
237:                     ),
238:     find_node(
239:           RefractiveIndex, Sodium, Magnesium, Aluminium,
240:           Silicon, Potassium, Calcium, Barium, Iron,
241:                   Node),
242:     find_category(Node, Cat),
243:     write('Category is '), write(Cat), nl.
244:
245: read_string(L):-
246:  get0(C), get_rest(C,L1), name(L,L1).
247: get_rest(10,[]):-
248:  !.
249: get_rest(13,[]):-
250:  !.
251: get_rest(C,[C|OUT]):-
252:   get0(C1), get_rest(C1,OUT).
253:
254: print_attributes(
```

```
255:           RefractiveIndex, Sodium, Magnesium, Aluminium,
256:           Silicon, Potassium, Calcium, Barium, Iron
257:                 ):-
258:     nl,
259:     write('Refractive index = '), write(RefractiveIndex), write('   '),nl,
260:     write('Sodium           = '), write(Sodium), write('   '),nl,
261:     write('Magnesium        = '), write(Magnesium), write('   '),nl,
262:     write('Aluminium        = '), write(Aluminium),write('   '),nl,
263:     write('Silicon          = '), write(Silicon), write('   '), nl,
264:     write('Potassium        = '), write(Potassium), write('   '),nl,
265:     write('Calcium          = '), write(Calcium), write('   '),nl,
266:     write('Barium           = '), write(Barium), write('   '),nl,
267:     write('Iron             = '), write(Iron), write('   '), nl
268: .
269:
270: find_node(RefractiveIndex, Sodium, Magnesium, Aluminium,
271:           Silicon, Potassium, Calcium, Barium, Iron
272:         , Node):-
273:     Magnesium < 2.7,
274:     Node = 1 .
275:
276: find_node(RefractiveIndex, Sodium, Magnesium, Aluminium,
277:           Silicon, Potassium, Calcium, Barium, Iron
278:         , Node):-
279:     Node is 1,
280:     RefractiveIndex < 1.52007,
281:     Node = 2 .
282:
283: find_node(RefractiveIndex, Sodium, Magnesium, Aluminium,
284:           Silicon, Potassium, Calcium, Barium, Iron
285:         , Node):-
286:     Node is 1,
287:     RefractiveIndex >= 1.52007,
288:     Node = 3 .
289:
290: find_node(RefractiveIndex, Sodium, Magnesium, Aluminium,
291:           Silicon, Potassium, Calcium, Barium, Iron
292:         , Node):-
293:     Node is 3,
294:     RefractiveIndex < 1.52516,
295:     Node = 4 .
296:
297: find_node(RefractiveIndex, Sodium, Magnesium, Aluminium,
298:           Silicon, Potassium, Calcium, Barium, Iron
299:         , Node):-
300:     Node is 4,
```

```
301:     Sodium < 13.495,
302:     Node = 5 .
303:
304: find_node(RefractiveIndex, Sodium, Magnesium, Aluminium,
305:           Silicon, Potassium, Calcium, Barium, Iron
306:         , Node):-
307:     Node is 4,
308:     Sodium >= 13.495,
309:     Node = 6 .
310:
311: find_node(RefractiveIndex, Sodium, Magnesium, Aluminium,
312:           Silicon, Potassium, Calcium, Barium, Iron
313:         , Node):-
314:     Node is 3,
315:     RefractiveIndex >= 1.52516,
316:     Node = 7 .
317:
318: find_node(RefractiveIndex, Sodium, Magnesium, Aluminium,
319:           Silicon, Potassium, Calcium, Barium, Iron
320:         , Node):-
321:     Magnesium >= 2.7,
322:     Node = 8 .
323:
324: find_node(RefractiveIndex, Sodium, Magnesium, Aluminium,
325:           Silicon, Potassium, Calcium, Barium, Iron
326:         , Node):-
327:     Node is 8,
328:     $Aluminium < 1.42,
329:     Node = 9 .
330:
331: find_node(RefractiveIndex, Sodium, Magnesium, Aluminium,
332:           Silicon, Potassium, Calcium, Barium, Iron
333:         , Node):-
334:     Node is 8,
335:     $Aluminium >= 1.42,
336:     Node = 10 .
337:
338: find_node(RefractiveIndex, Sodium, Magnesium, Aluminium,
339:           Silicon, Potassium, Calcium, Barium, Iron
340:         , Node):-
341:     Node is 9,
342:     $Refractiveindex < 1.51707,
343:     Node = 11 .
344:
345: find_node(RefractiveIndex, Sodium, Magnesium, Aluminium,
346:           Silicon, Potassium, Calcium, Barium, Iron
```

```
347:         , Node):-
348:     Node is 9,
349:     $Refractiveindex >= 1.51707,
350:     Node = 12 .
351:
352: find_node(RefractiveIndex, Sodium, Magnesium, Aluminium,
353:           Silicon, Potassium, Calcium, Barium, Iron
354:         , Node):-
355:     Node is 11,
356:     $Refractiveindex < 1.51599,
357:     Node = 13 .
358:
359: find_node(RefractiveIndex, Sodium, Magnesium, Aluminium,
360:           Silicon, Potassium, Calcium, Barium, Iron
361:         , Node):-
362:     Node is 11,
363:     $Refractiveindex >= 1.51599,
364:     Node = 14 .
365:
366: %
367: % find_category predicates.
368: %
369: find_category(Node, Cat):-
370:     Node is 7,
371:     Cat = 2 .
372:
373: find_category(Node, Cat):-
374:     Node is 13,
375:     Cat = 1 .
376:
377: find_category(Node, Cat):-
378:     Node is 14,
379:     Cat = 3 .
380:
381: find_category(Node, Cat):-
382:     Node is 0,
383:     Cat = 0 .
384:
385: find_category(Node, Cat):-
386:     Node is 2,
387:     Cat = 7 .
388:
389: find_category(Node, Cat):-
390:     Node is 5,
391:     Cat = 5 .
392:
```

```
393: find_category(Node, Cat):-
394:     Node is 6,
395:     Cat = 2 .
396:
```

C.9.3 glass05.pro

```
220: %
221: % Here are the classification rules
222: %
223: go:-
224:     Cat = 0,
225:     identify(Gid).
226:
227: identify(Gid):-
228:     write('Enter glass id: '),
229:     read_string(Gid),
230:     glass(Gid,
231:           RefractiveIndex, Sodium, Magnesium, Aluminium,
232:           Silicon, Potassium, Calcium, Barium, Iron
233:           ),
234:     print_attributes(
235:           RefractiveIndex, Sodium, Magnesium, Aluminium,
236:           Silicon, Potassium, Calcium, Barium, Iron
237:                     ),
238:     find_category(
239:           RefractiveIndex, Sodium, Magnesium, Aluminium,
240:           Silicon, Potassium, Calcium, Barium, Iron,
241:                   Cat),
242:     write('Category is '), write(Cat), nl.
243:
244: read_string(L):-
245:  get0(C), get_rest(C,L1), name(L,L1).
246: get_rest(10,[]):-
247:  !.
248: get_rest(13,[]):-
249:  !.
250: get_rest(C,[C|OUT]):-
251:   get0(C1), get_rest(C1,OUT).
252:
253: print_attributes(
254:           RefractiveIndex, Sodium, Magnesium, Aluminium,
255:           Silicon, Potassium, Calcium, Barium, Iron
256:                 ):-
257:     nl,
258:     write('Refractive index = '), write(RefractiveIndex), write('   '),nl,
```

```
259:     write('Sodium          = '), write(Sodium), write('   '),nl,
260:     write('Magnesium       = '), write(Magnesium), write('   '),nl,
261:     write('Aluminium       = '), write(Aluminium),write('   '),nl,
262:     write('Silicon         = '), write(Silicon), write('   '), nl,
263:     write('Potassium       = '), write(Potassium), write('   '),nl,
264:     write('Calcium         = '), write(Calcium), write('   '),nl,
265:     write('Barium          = '), write(Barium), write('   '),nl,
266:     write('Iron            = '), write(Iron), write('   '), nl
267: .
268:
269: find_category(RefractiveIndex, Sodium, Magnesium, Aluminium,
270:           Silicon, Potassium, Calcium, Barium, Iron
271:         , Cat):-
272: Barium > 0.27,
273: test1Si(RefractiveIndex, Sodium, Magnesium, Aluminium,
274:           Silicon, Potassium, Calcium, Barium, Iron, Cat)
275: .
276:
277: find_category(RefractiveIndex, Sodium, Magnesium, Aluminium,
278:           Silicon, Potassium, Calcium, Barium, Iron
279:         , Cat):-
280: Barium =< 0.27,
281: write('about to testmg'),nl,
282: testMg(RefractiveIndex, Sodium, Magnesium, Aluminium,
283:           Silicon, Potassium, Calcium, Barium, Iron, Cat)
284: .
285:
286: test1Si(RefractiveIndex, Sodium, Magnesium, Aluminium,
287:           Silicon, Potassium, Calcium, Barium, Iron, Cat):-
288: Silicon =< 70.16,
289: Cat = 2
290: .
291:
292: test1Si(RefractiveIndex, Sodium, Magnesium, Aluminium,
293:           Silicon, Potassium, Calcium, Barium, Iron, Cat):-
294: Silicon > 70.16,
295: Cat = 7
296: .
297:
298: testMg(RefractiveIndex, Sodium, Magnesium, Aluminium,
299:           Silicon, Potassium, Calcium, Barium, Iron, Cat):-
300: Magnesium =< 2.41,
301: write('will test k'),nl,
302: testK(RefractiveIndex, Sodium, Magnesium, Aluminium,
303:           Silicon, Potassium, Calcium, Barium, Iron, Cat)
304: .
```

```
305:
306: testMg(RefractiveIndex, Sodium, Magnesium, Aluminium,
307:           Silicon, Potassium, Calcium, Barium, Iron, Cat):-
308: Magnesium > 2.41,
309: testAl(RefractiveIndex, Sodium, Magnesium, Aluminium,
310:           Silicon, Potassium, Calcium, Barium, Iron, Cat)
311: .
312:
313: testK(RefractiveIndex, Sodium, Magnesium, Aluminium,
314:           Silicon, Potassium, Calcium, Barium, Iron, Cat):-
315: Potassium =< 0.03,
316: write('potassium is less than 0.03'),nl,
317: test1Na(RefractiveIndex, Sodium, Magnesium, Aluminium,
318:           Silicon, Potassium, Calcium, Barium, Iron, Cat)
319: .
320:
321: testK(RefractiveIndex, Sodium, Magnesium, Aluminium,
322:           Silicon, Potassium, Calcium, Barium, Iron, Cat):-
323: Potassium > 0.03,
324: write('potassium is more than 0.03'),nl,
325: test2Na(RefractiveIndex, Sodium, Magnesium, Aluminium,
326:           Silicon, Potassium, Calcium, Barium, Iron, Cat)
327: .
328:
329: test1Na(RefractiveIndex, Sodium, Magnesium, Aluminium,
330:           Silicon, Potassium, Calcium, Barium, Iron, Cat):-
331: Sodium =< 13.75,
332: Cat = 2
333: .
334:
335: test1Na(RefractiveIndex, Sodium, Magnesium, Aluminium,
336:           Silicon, Potassium, Calcium, Barium, Iron, Cat):-
337: Sodium > 13.75,
338: Cat = 6
339: .
340:
341: test2Na(RefractiveIndex, Sodium, Magnesium, Aluminium,
342:           Silicon, Potassium, Calcium, Barium, Iron, Cat):-
343: Sodium =< 13.49,
344: testRI(RefractiveIndex, Sodium, Magnesium, Aluminium,
345:           Silicon, Potassium, Calcium, Barium, Iron, Cat)
346: .
347:
348: test2Na(RefractiveIndex, Sodium, Magnesium, Aluminium,
349:           Silicon, Potassium, Calcium, Barium, Iron, Cat):-
350: Sodium > 13.49,
```

```
351: Cat = 2
352: .
353:
354: testRI(RefractiveIndex, Sodium, Magnesium, Aluminium,
355:           Silicon, Potassium, Calcium, Barium, Iron, Cat):-
356: RefractiveIndex =< 1.5241,
357: Cat = 5
358: .
359:
360: testRI(RefractiveIndex, Sodium, Magnesium, Aluminium,
361:           Silicon, Potassium, Calcium, Barium, Iron, Cat):-
362: RefractiveIndex > 1.5241,
363: Cat = 2
364: .
365:
366: testAl(RefractiveIndex, Sodium, Magnesium, Aluminium,
367:           Silicon, Potassium, Calcium, Barium, Iron, Cat):-
368: write('entered test al'),nl,
369: Aluminium =< 1.41,
370: test2RI(RefractiveIndex, Sodium, Magnesium, Aluminium,
371:           Silicon, Potassium, Calcium, Barium, Iron, Cat)
372: .
373:
374: testAl(RefractiveIndex, Sodium, Magnesium, Aluminium,
375:           Silicon, Potassium, Calcium, Barium, Iron, Cat):-
376: Aluminium > 1.41,
377: test2Si(RefractiveIndex, Sodium, Magnesium, Aluminium,
378:           Silicon, Potassium, Calcium, Barium, Iron, Cat)
379: .
380:
381: test2RI(RefractiveIndex, Sodium, Magnesium, Aluminium,
382:           Silicon, Potassium, Calcium, Barium, Iron, Cat):-
383: RefractiveIndex =< 1.51707,
384: test3RI(RefractiveIndex, Sodium, Magnesium, Aluminium,
385:           Silicon, Potassium, Calcium, Barium, Iron, Cat)
386: .
387:
388: test2RI(RefractiveIndex, Sodium, Magnesium, Aluminium,
389:           Silicon, Potassium, Calcium, Barium, Iron, Cat):-
390: RefractiveIndex > 1.51707,
391: write('about to test3k'),nl,
392: test3K(RefractiveIndex, Sodium, Magnesium, Aluminium,
393:           Silicon, Potassium, Calcium, Barium, Iron, Cat)
394: .
395:
396: test3RI(RefractiveIndex, Sodium, Magnesium, Aluminium,
```

```
397:           Silicon, Potassium, Calcium, Barium, Iron, Cat):-
398: RefractiveIndex =< 1.51596,
399: Cat = 1
400: .
401:
402: test3RI(RefractiveIndex, Sodium, Magnesium, Aluminium,
403:           Silicon, Potassium, Calcium, Barium, Iron, Cat):-
404: RefractiveIndex > 1.51596,
405: test1Fe(RefractiveIndex, Sodium, Magnesium, Aluminium,
406:           Silicon, Potassium, Calcium, Barium, Iron, Cat)
407: .
408:
409: test1Fe(RefractiveIndex, Sodium, Magnesium, Aluminium,
410:           Silicon, Potassium, Calcium, Barium, Iron, Cat):-
411: Iron =< 0.12,
412: test2Al(RefractiveIndex, Sodium, Magnesium, Aluminium,
413:           Silicon, Potassium, Calcium, Barium, Iron, Cat)
414: .
415:
416: test1Fe(RefractiveIndex, Sodium, Magnesium, Aluminium,
417:           Silicon, Potassium, Calcium, Barium, Iron, Cat):-
418: Iron > 0.12,
419: Cat = 2
420: .
421:
422: test2Al(RefractiveIndex, Sodium, Magnesium, Aluminium,
423:           Silicon, Potassium, Calcium, Barium, Iron, Cat):-
424: Aluminium =< 1.27,
425: test2Mg(RefractiveIndex, Sodium, Magnesium, Aluminium,
426:           Silicon, Potassium, Calcium, Barium, Iron, Cat)
427: .
428:
429:
430: test2Al(RefractiveIndex, Sodium, Magnesium, Aluminium,
431:           Silicon, Potassium, Calcium, Barium, Iron, Cat):-
432: Aluminium > 1.27,
433: Cat = 3
434: .
435:
436: test2Mg(RefractiveIndex, Sodium, Magnesium, Aluminium,
437:           Silicon, Potassium, Calcium, Barium, Iron, Cat):-
438: Magnesium =< 3.47,
439: Cat = 3
440: .
441:
442: test2Mg(RefractiveIndex, Sodium, Magnesium, Aluminium,
```

```
443:           Silicon, Potassium, Calcium, Barium, Iron, Cat):-
444: Magnesium > 3.47,
445: Cat = 2
446: .
447:
448: test3K(RefractiveIndex, Sodium, Magnesium, Aluminium,
449:           Silicon, Potassium, Calcium, Barium, Iron, Cat):-
450: Potassium =< 0.23,
451: test3Mg(RefractiveIndex, Sodium, Magnesium, Aluminium,
452:           Silicon, Potassium, Calcium, Barium, Iron, Cat)
453: .
454:
455: test3K(RefractiveIndex, Sodium, Magnesium, Aluminium,
456:           Silicon, Potassium, Calcium, Barium, Iron, Cat):-
457: Potassium > 0.23,
458: test4Mg(RefractiveIndex, Sodium, Magnesium, Aluminium,
459:           Silicon, Potassium, Calcium, Barium, Iron, Cat)
460: .
461:
462: test3Mg(RefractiveIndex, Sodium, Magnesium, Aluminium,
463:           Silicon, Potassium, Calcium, Barium, Iron, Cat):-
464: Magnesium =< 3.34,
465: Cat = 2
466: .
467:
468: test3Mg(RefractiveIndex, Sodium, Magnesium, Aluminium,
469:           Silicon, Potassium, Calcium, Barium, Iron, Cat):-
470: Magnesium > 3.34,
471: test2Si(RefractiveIndex, Sodium, Magnesium, Aluminium,
472:           Silicon, Potassium, Calcium, Barium, Iron, Cat)
473: .
474:
475: test2Si(RefractiveIndex, Sodium, Magnesium, Aluminium,
476:           Silicon, Potassium, Calcium, Barium, Iron, Cat):-
477: Silicon =< 72.64,
478: Cat = 1
479: .
480:
481: test2Si(RefractiveIndex, Sodium, Magnesium, Aluminium,
482:           Silicon, Potassium, Calcium, Barium, Iron, Cat):-
483: Silicon > 72.64,
484: Cat = 3
485: .
486:
487: test4Mg(RefractiveIndex, Sodium, Magnesium, Aluminium,
488:           Silicon, Potassium, Calcium, Barium, Iron, Cat):-
```

```
489: Magnesium =< 3.75,
490: test2Fe(RefractiveIndex, Sodium, Magnesium, Aluminium,
491:           Silicon, Potassium, Calcium, Barium, Iron, Cat)
492: .
493:
494: test4Mg(RefractiveIndex, Sodium, Magnesium, Aluminium,
495:           Silicon, Potassium, Calcium, Barium, Iron, Cat):-
496: Magnesium > 3.75,
497: Cat = 2
498: .
499:
500: test2Fe(RefractiveIndex, Sodium, Magnesium, Aluminium,
501:           Silicon, Potassium, Calcium, Barium, Iron, Cat):-
502: Iron =< 0.14,
503: test4RI(RefractiveIndex, Sodium, Magnesium, Aluminium,
504:           Silicon, Potassium, Calcium, Barium, Iron, Cat)
505: .
506:
507: test2Fe(RefractiveIndex, Sodium, Magnesium, Aluminium,
508:           Silicon, Potassium, Calcium, Barium, Iron, Cat):-
509: Iron > 0.14,
510: test3Al(RefractiveIndex, Sodium, Magnesium, Aluminium,
511:           Silicon, Potassium, Calcium, Barium, Iron, Cat)
512: .
513:
514: test4RI(RefractiveIndex, Sodium, Magnesium, Aluminium,
515:           Silicon, Potassium, Calcium, Barium, Iron, Cat):-
516: RefractiveIndex =< 1.52043,
517: Cat = 1
518: .
519:
520: test4RI(RefractiveIndex, Sodium, Magnesium, Aluminium,
521:           Silicon, Potassium, Calcium, Barium, Iron, Cat):-
522: RefractiveIndex > 1.52043,
523: Cat = 2
524: .
525:
526: test3Al(RefractiveIndex, Sodium, Magnesium, Aluminium,
527:           Silicon, Potassium, Calcium, Barium, Iron, Cat):-
528: Aluminium =< 1.17,
529: Cat = 2
530: .
531:
532: test3Al(RefractiveIndex, Sodium, Magnesium, Aluminium,
533:           Silicon, Potassium, Calcium, Barium, Iron, Cat):-
534: Aluminium > 1.17,
```

```
535: Cat = 1
536: .
537:
538: test2Si(RefractiveIndex, Sodium, Magnesium, Aluminium,
539:           Silicon, Potassium, Calcium, Barium, Iron, Cat):-
540: Silicon =< 72.49,
541: test1Ca(RefractiveIndex, Sodium, Magnesium, Aluminium,
542:           Silicon, Potassium, Calcium, Barium, Iron, Cat)
543: .
544:
545: test2Si(RefractiveIndex, Sodium, Magnesium, Aluminium,
546:           Silicon, Potassium, Calcium, Barium, Iron, Cat):-
547: Silicon > 72.49,
548: Cat = 2
549: .
550:
551: test1Ca(RefractiveIndex, Sodium, Magnesium, Aluminium,
552:           Silicon, Potassium, Calcium, Barium, Iron, Cat):-
553: Calcium =< 8.28,
554: Cat = 2
555: .
556:
557: test1Ca(RefractiveIndex, Sodium, Magnesium, Aluminium,
558:           Silicon, Potassium, Calcium, Barium, Iron, Cat):-
559: Calcium > 8.28,
560: Cat = 3
561: .
```

C.9.4 glass06.pro

Note that for brevity, the fact statements are omitted. They are the same as for `glass05.pro` except that the category is the last item for each fact statement.

```
220: %
221: % Here are the classification rules
222: %
223: go:-
224:     Cat = 0,
225:     Match = 0,
226:     identify(Gid).
227:
228: identify(Gid):-
229:     write('Enter glass id: '),
230:     read_string(Gid),
231:     glass(Gid,
232:           RefractiveIndex, Sodium, Magnesium, Aluminium,
```

```
233:           Silicon, Potassium, Calcium, Barium, Iron, Category
234:           ),
235:     find_category(
236:           RefractiveIndex, Sodium, Magnesium, Aluminium,
237:           Silicon, Potassium, Calcium, Barium, Iron,
238:                   Cat),
239:     find_match(Cat, Category, Match),
240:     write('id '), write(Gid),
241:     write(' match = '), write(Match), nl
242: .
243:
244: read_string(L):-
245:  get0(C), get_rest(C,L1), name(L,L1).
246: get_rest(10,[]):-
247:  !.
248: get_rest(13,[]):-
249:  !.
250: get_rest(C,[C|OUT]):-
251:   get0(C1), get_rest(C1,OUT).
252:
253: find_category(RefractiveIndex, Sodium, Magnesium, Aluminium,
254:           Silicon, Potassium, Calcium, Barium, Iron
255:         , Cat):-
256: Barium > 0.27,
257: test1Si(RefractiveIndex, Sodium, Magnesium, Aluminium,
258:           Silicon, Potassium, Calcium, Barium, Iron, Cat)
259: .
260:
261: find_category(RefractiveIndex, Sodium, Magnesium, Aluminium,
262:           Silicon, Potassium, Calcium, Barium, Iron
263:         , Cat):-
264: Barium =< 0.27,
265: testMg(RefractiveIndex, Sodium, Magnesium, Aluminium,
266:           Silicon, Potassium, Calcium, Barium, Iron, Cat)
267: .
268:
269: test1Si(RefractiveIndex, Sodium, Magnesium, Aluminium,
270:           Silicon, Potassium, Calcium, Barium, Iron, Cat):-
271: Silicon =< 70.16,
272: Cat = 2
273: .
274:
275: test1Si(RefractiveIndex, Sodium, Magnesium, Aluminium,
276:           Silicon, Potassium, Calcium, Barium, Iron, Cat):-
277: Silicon > 70.16,
278: Cat = 7
```

```
279: .
280:
281: testMg(RefractiveIndex, Sodium, Magnesium, Aluminium,
282:           Silicon, Potassium, Calcium, Barium, Iron, Cat):-
283: Magnesium =< 2.41,
284: testK(RefractiveIndex, Sodium, Magnesium, Aluminium,
285:           Silicon, Potassium, Calcium, Barium, Iron, Cat)
286: .
287:
288: testMg(RefractiveIndex, Sodium, Magnesium, Aluminium,
289:           Silicon, Potassium, Calcium, Barium, Iron, Cat):-
290: Magnesium > 2.41,
291: testAl(RefractiveIndex, Sodium, Magnesium, Aluminium,
292:           Silicon, Potassium, Calcium, Barium, Iron, Cat)
293: .
294:
295: testK(RefractiveIndex, Sodium, Magnesium, Aluminium,
296:           Silicon, Potassium, Calcium, Barium, Iron, Cat):-
297: Potassium =< 0.03,
298: test1Na(RefractiveIndex, Sodium, Magnesium, Aluminium,
299:           Silicon, Potassium, Calcium, Barium, Iron, Cat)
300: .
301:
302: testK(RefractiveIndex, Sodium, Magnesium, Aluminium,
303:           Silicon, Potassium, Calcium, Barium, Iron, Cat):-
304: Potassium > 0.03,
305: test2Na(RefractiveIndex, Sodium, Magnesium, Aluminium,
306:           Silicon, Potassium, Calcium, Barium, Iron, Cat)
307: .
308:
309: test1Na(RefractiveIndex, Sodium, Magnesium, Aluminium,
310:           Silicon, Potassium, Calcium, Barium, Iron, Cat):-
311: Sodium =< 13.75,
312: Cat = 2
313: .
314:
315: test1Na(RefractiveIndex, Sodium, Magnesium, Aluminium,
316:           Silicon, Potassium, Calcium, Barium, Iron, Cat):-
317: Sodium > 13.75,
318: Cat = 6
319: .
320:
321: test2Na(RefractiveIndex, Sodium, Magnesium, Aluminium,
322:           Silicon, Potassium, Calcium, Barium, Iron, Cat):-
323: Sodium =< 13.49,
324: testRI(RefractiveIndex, Sodium, Magnesium, Aluminium,
```

```
325:           Silicon, Potassium, Calcium, Barium, Iron, Cat)
326: .
327:
328: test2Na(RefractiveIndex, Sodium, Magnesium, Aluminium,
329:           Silicon, Potassium, Calcium, Barium, Iron, Cat):-
330: Sodium > 13.49,
331: Cat = 2
332: .
333:
334: testRI(RefractiveIndex, Sodium, Magnesium, Aluminium,
335:           Silicon, Potassium, Calcium, Barium, Iron, Cat):-
336: RefractiveIndex =< 1.5241,
337: Cat = 5
338: .
339:
340: testRI(RefractiveIndex, Sodium, Magnesium, Aluminium,
341:           Silicon, Potassium, Calcium, Barium, Iron, Cat):-
342: RefractiveIndex > 1.5241,
343: Cat = 2
344: .
345:
346: testAl(RefractiveIndex, Sodium, Magnesium, Aluminium,
347:           Silicon, Potassium, Calcium, Barium, Iron, Cat):-
348: Aluminium =< 1.41,
349: test2RI(RefractiveIndex, Sodium, Magnesium, Aluminium,
350:           Silicon, Potassium, Calcium, Barium, Iron, Cat)
351: .
352:
353: testAl(RefractiveIndex, Sodium, Magnesium, Aluminium,
354:           Silicon, Potassium, Calcium, Barium, Iron, Cat):-
355: Aluminium > 1.41,
356: test2Si(RefractiveIndex, Sodium, Magnesium, Aluminium,
357:           Silicon, Potassium, Calcium, Barium, Iron, Cat)
358: .
359:
360: test2RI(RefractiveIndex, Sodium, Magnesium, Aluminium,
361:           Silicon, Potassium, Calcium, Barium, Iron, Cat):-
362: RefractiveIndex =< 1.51707,
363: test3RI(RefractiveIndex, Sodium, Magnesium, Aluminium,
364:           Silicon, Potassium, Calcium, Barium, Iron, Cat)
365: .
366:
367: test2RI(RefractiveIndex, Sodium, Magnesium, Aluminium,
368:           Silicon, Potassium, Calcium, Barium, Iron, Cat):-
369: RefractiveIndex > 1.51707,
370: test3K(RefractiveIndex, Sodium, Magnesium, Aluminium,
```

```
371:           Silicon, Potassium, Calcium, Barium, Iron, Cat)
372: .
373:
374: test3RI(RefractiveIndex, Sodium, Magnesium, Aluminium,
375:           Silicon, Potassium, Calcium, Barium, Iron, Cat):-
376: RefractiveIndex =< 1.51596,
377: Cat = 1
378: .
379:
380: test3RI(RefractiveIndex, Sodium, Magnesium, Aluminium,
381:           Silicon, Potassium, Calcium, Barium, Iron, Cat):-
382: RefractiveIndex > 1.51596,
383: test1Fe(RefractiveIndex, Sodium, Magnesium, Aluminium,
384:           Silicon, Potassium, Calcium, Barium, Iron, Cat)
385: .
386:
387: test1Fe(RefractiveIndex, Sodium, Magnesium, Aluminium,
388:           Silicon, Potassium, Calcium, Barium, Iron, Cat):-
389: Iron =< 0.12,
390: test2Al(RefractiveIndex, Sodium, Magnesium, Aluminium,
391:           Silicon, Potassium, Calcium, Barium, Iron, Cat)
392: .
393:
394: test1Fe(RefractiveIndex, Sodium, Magnesium, Aluminium,
395:           Silicon, Potassium, Calcium, Barium, Iron, Cat):-
396: Iron > 0.12,
397: Cat = 2
398: .
399:
400: test2Al(RefractiveIndex, Sodium, Magnesium, Aluminium,
401:           Silicon, Potassium, Calcium, Barium, Iron, Cat):-
402: Aluminium =< 1.27,
403: test2Mg(RefractiveIndex, Sodium, Magnesium, Aluminium,
404:           Silicon, Potassium, Calcium, Barium, Iron, Cat)
405: .
406:
407:
408: test2Al(RefractiveIndex, Sodium, Magnesium, Aluminium,
409:           Silicon, Potassium, Calcium, Barium, Iron, Cat):-
410: Aluminium > 1.27,
411: Cat = 3
412: .
413:
414: test2Mg(RefractiveIndex, Sodium, Magnesium, Aluminium,
415:           Silicon, Potassium, Calcium, Barium, Iron, Cat):-
416: Magnesium =< 3.47,
```

```
417: Cat = 3
418: .
419:
420: test2Mg(RefractiveIndex, Sodium, Magnesium, Aluminium,
421:           Silicon, Potassium, Calcium, Barium, Iron, Cat):-
422: Magnesium > 3.47,
423: Cat = 2
424: .
425:
426: test3K(RefractiveIndex, Sodium, Magnesium, Aluminium,
427:           Silicon, Potassium, Calcium, Barium, Iron, Cat):-
428: Potassium =< 0.23,
429: test3Mg(RefractiveIndex, Sodium, Magnesium, Aluminium,
430:           Silicon, Potassium, Calcium, Barium, Iron, Cat)
431: .
432:
433: test3K(RefractiveIndex, Sodium, Magnesium, Aluminium,
434:           Silicon, Potassium, Calcium, Barium, Iron, Cat):-
435: Potassium > 0.23,
436: test4Mg(RefractiveIndex, Sodium, Magnesium, Aluminium,
437:           Silicon, Potassium, Calcium, Barium, Iron, Cat)
438: .
439:
440: test3Mg(RefractiveIndex, Sodium, Magnesium, Aluminium,
441:           Silicon, Potassium, Calcium, Barium, Iron, Cat):-
442: Magnesium =< 3.34,
443: Cat = 2
444: .
445:
446: test3Mg(RefractiveIndex, Sodium, Magnesium, Aluminium,
447:           Silicon, Potassium, Calcium, Barium, Iron, Cat):-
448: Magnesium > 3.34,
449: test2Si(RefractiveIndex, Sodium, Magnesium, Aluminium,
450:           Silicon, Potassium, Calcium, Barium, Iron, Cat)
451: .
452:
453: test2Si(RefractiveIndex, Sodium, Magnesium, Aluminium,
454:           Silicon, Potassium, Calcium, Barium, Iron, Cat):-
455: Silicon =< 72.64,
456: Cat = 1
457: .
458:
459: test2Si(RefractiveIndex, Sodium, Magnesium, Aluminium,
460:           Silicon, Potassium, Calcium, Barium, Iron, Cat):-
461: Silicon > 72.64,
462: Cat = 3
```

```
463: .
464:
465: test4Mg(RefractiveIndex, Sodium, Magnesium, Aluminium,
466:           Silicon, Potassium, Calcium, Barium, Iron, Cat):-
467: Magnesium =< 3.75,
468: test2Fe(RefractiveIndex, Sodium, Magnesium, Aluminium,
469:           Silicon, Potassium, Calcium, Barium, Iron, Cat)
470: .
471:
472: test4Mg(RefractiveIndex, Sodium, Magnesium, Aluminium,
473:           Silicon, Potassium, Calcium, Barium, Iron, Cat):-
474: Magnesium > 3.75,
475: Cat = 2
476: .
477:
478: test2Fe(RefractiveIndex, Sodium, Magnesium, Aluminium,
479:           Silicon, Potassium, Calcium, Barium, Iron, Cat):-
480: Iron =< 0.14,
481: test4RI(RefractiveIndex, Sodium, Magnesium, Aluminium,
482:           Silicon, Potassium, Calcium, Barium, Iron, Cat)
483: .
484:
485: test2Fe(RefractiveIndex, Sodium, Magnesium, Aluminium,
486:           Silicon, Potassium, Calcium, Barium, Iron, Cat):-
487: Iron > 0.14,
488: test3Al(RefractiveIndex, Sodium, Magnesium, Aluminium,
489:           Silicon, Potassium, Calcium, Barium, Iron, Cat)
490: .
491:
492: test4RI(RefractiveIndex, Sodium, Magnesium, Aluminium,
493:           Silicon, Potassium, Calcium, Barium, Iron, Cat):-
494: RefractiveIndex =< 1.52043,
495: Cat = 1
496: .
497:
498: test4RI(RefractiveIndex, Sodium, Magnesium, Aluminium,
499:           Silicon, Potassium, Calcium, Barium, Iron, Cat):-
500: RefractiveIndex > 1.52043,
501: Cat = 2
502: .
503:
504: test3Al(RefractiveIndex, Sodium, Magnesium, Aluminium,
505:           Silicon, Potassium, Calcium, Barium, Iron, Cat):-
506: Aluminium =< 1.17,
507: Cat = 2
508: .
```

```
509:
510: test3Al(RefractiveIndex, Sodium, Magnesium, Aluminium,
511:           Silicon, Potassium, Calcium, Barium, Iron, Cat):-
512: Aluminium > 1.17,
513: Cat = 1
514: .
515:
516: test2Si(RefractiveIndex, Sodium, Magnesium, Aluminium,
517:           Silicon, Potassium, Calcium, Barium, Iron, Cat):-
518: Silicon =< 72.49,
519: test1Ca(RefractiveIndex, Sodium, Magnesium, Aluminium,
520:           Silicon, Potassium, Calcium, Barium, Iron, Cat)
521: .
522:
523: test2Si(RefractiveIndex, Sodium, Magnesium, Aluminium,
524:           Silicon, Potassium, Calcium, Barium, Iron, Cat):-
525: Silicon > 72.49,
526: Cat = 2
527: .
528:
529: test1Ca(RefractiveIndex, Sodium, Magnesium, Aluminium,
530:           Silicon, Potassium, Calcium, Barium, Iron, Cat):-
531: Calcium =< 8.28,
532: Cat = 2
533: .
534:
535: test1Ca(RefractiveIndex, Sodium, Magnesium, Aluminium,
536:           Silicon, Potassium, Calcium, Barium, Iron, Cat):-
537: Calcium > 8.28,
538: Cat = 3
539: .
540:
541: find_match(Cat, Category, Match):-
542: Cat is Category,
543: Match = 1
544: .
545:
546: find_match(Cat, Category, Match):-
547: Match is 0,
548: Match = 0
549: .
```

C.10 marry.pl

```
1: #! /bin/perl
```

```
 2:
 3: # this is the perl script that marries the neural network
 4: # with the knowledge base work.
 5:
 6: use strict;
 7:
 8: sub create_automate
 9: {
10:     my $rootname;
11:     my $filename;
12:     my $nnfile;
13:
14:     $rootname = $_[0];
15:     $nnfile = $rootname.".nna";
16:
17:     $filename = "c:\\everything\\andrew\\ou\\t396\\ounw\\program\\automate.dat";
18:
19:     open(FD, ">".$filename) || die("Failed to open $filename for writing.");
20:     print FD "open ".$rootname.".nnd\n".
21: "initialize\n".
22: "set learnfile ".$nnfile."\n".
23: "learn 50000\n".
24: "test\n";
25:     close(FD);
26:
27:     print "Auomate file created ok.\n";
28: }
29:
30: sub run_neuralworks
31: {
32:     my $rootname;
33:     my $cmd;
34:
35:     $rootname = $_[0];
36:
37:      # Clear out old output file first.
38:     unlink($rootname.".nnr");
39:
40:     print "About to run NN\n";
41:     $cmd = "ounw -xuautomate";
42:     system($cmd);
43:     print "NN run completed.\n";
44: }
45:
46: sub get_record
47: {
```

```
48:     my $rootname;
49:     my $recnum;
50:     my $i;
51:     my $record;
52:     my $filename;
53:
54:     $rootname = $_[0];
55:     $recnum = $_[1];
56:
57:     $filename = $rootname.".nnr";
58:     open(FD, $filename) || die("Unable to open results file '".$filename."'.\n");
59:     $i = 0;
60:
61:     while(<FD>)
62:     {
63:         $record = $_;
64:         chomp($record);
65:         $i = $i + 1;
66:         if ($i == $recnum)
67:         {
68:             return $record;
69:         }
70:     }
71:     close(FD);
72:     return "";
73: }
74:
75: sub is_match
76: {
77:     my $record;
78:     my $outputs;
79:
80:     my $match;
81:     my @values;
82:     my $onendx;
83:     my $maxndx;
84:     my $maxval;
85:     my $i;
86:     my $out2;
87:
88:     $record = $_[0];
89:     $outputs = $_[1];
90:
91:     $match = 0;
92:     @values = split(/\s+/, $record);
93:     $onendx = 0;
```

```
 94:     for ($i = 0; $i < $outputs; $i = $i + 1)
 95:     {
 96:         if ($values[$i] == 1)
 97:         {
 98:             $onendx = $i;
 99:         }
100:     }
101:
102:     $maxndx = $outputs;
103:     $maxval = $values[$outputs];
104:     $out2 = $outputs * 2;
105:     for ($i = $outputs; $i < $out2; $i = $i + 1)
106:     {
107:         if ($values[$i] >= $values[$maxndx])
108:         {
109:             $maxndx = $i;
110:             $maxval = $values[$i];
111:         }
112:     }
113:
114:     $maxndx = $maxndx - $outputs;
115:     if ($onendx == $maxndx)
116:     {
117:         $match = 1;
118:     }
119:
120:     return $match;
121: }
122:
123: sub create_prolog_runner
124: {
125:     my $filename;
126:     my $glassid;
127:     my $batchfilename;
128:     my $quote;
129:     my $line;
130:
131:     $filename = $_[0];
132:     $glassid = $_[1];
133:
134:     $quote = '"';
135:     $batchfilename = "run_prolog.bat";
136:     unlink($batchfilename);
137:     open (FD, ">".$batchfilename) ||
138:          die("Failed to create PROLOG batch file.\n");
139:     print FD "c:\\everything\\andrew\\ou\\tm426\\swiProlog\\pl\\bin\\plcon < g5.da
```

```
140:     close(FD);
141:
142:     open (FD2, ">g5.dat") ||
143:          die("cannot create stdin file (g5.dat) for g5.bat");
144:     print FD2 "['glass08.pl'].\n";
145:     print FD2 "go.\n";
146:     close(FD2);
147:
148:     print FD2 "go:-\n";
149:     print FD2 "  Cat = 0,\n";
150:     print FD2 "  Gid = $glassid,\n";
151:     print FD2 "  identify(Gid).\n";
152:
153:     # create a pl file with the constant stuff in it
154:     # and the rule to run it with the specified glass id.
155:
156:     open (FD3, ">glass08.pl") ||
157:          die("cannot create glass08.pl");
158:
159:     print FD3 "go:-\n";
160:     print FD3 "  Cat = 0,\n";
161:     print FD3 "  Match = 0,\n";
162:     print FD3 "  Gid = $glassid,\n";
163:     print FD3 "  identify(Gid).\n\n";
164:
165:     open (FD3b, "glass06b.pro") ||
166:         die ("cannot open glass06b.pro for reading");
167:     while (<FD3b>)
168:     {
169:         $line = $_;
170:         print FD3 $line;
171:     }
172:     close (FD3b);
173:
174:     close (FD3);
175: }
176:
177: sub run_prolog
178: {
179:     my $cmd;
180:     my $filename;
181:     my $glassid;
182:     my $cmd;
183:
184:     $filename = $_[0];
185:     $glassid = $_[1];
```

```
186:
187:     chdir("c:\\everything\\andrew\\ou\\tm426");
188:     create_prolog_runner($filename, $glassid);
189:     $cmd = "run_prolog.bat";
190:     system($cmd);
191: }
192:
193: sub main
194: {
195:     my $glassid;
196:     my $record;
197:     my $rootname;
198:     my $nn_match;
199:
200:     if (scalar(@ARGV) == 0)
201:     {
202:         print STDERR "Usage: marry.pl <NN-file-prefix>\n";
203:         exit(1);
204:     }
205:
206:     $rootname = $ARGV[0];
207:
208:     chdir("c:\\everything\\andrew\\ou\\t396\\ounw\\program");
209:
210:     create_automate($rootname);
211:     run_neuralworks($rootname);
212:
213:     chdir("c:\\everything\\andrew\\ou\\t396\\ounw");
214:
215:     for ($glassid = 1; $glassid <= 201; $glassid++)
216:     {
217:         $record = get_record($rootname, $glassid);
218:         $nn_match = is_match($record, 7);
219:         if ($nn_match)
220:         {
221:             print "The neural network answer is correct for glassid $glassid, matc
222:         }
223:         else
224:         {
225:             print "The neural network answer is wrong.\n";
226:             print "Will now try using prolog decision tree.\n";
227:             chdir("c:\\everything\\andrew\\ou\\tm426\\software");
228:             run_prolog("g5.pl", $glassid);
229:             print "Prolog run finished.\n";
230:         }
231:     }
```

```
232: }
233:
234: main();
```

Bibliography

[1] Gennady Agre and Irena Kopinska. *Case-based refinement of Knowledge-based Neural Networks.* Institute of Information Technologies, Bulgarian Academy of Science, 1995.

[2] C.L. Blake and C.J. Merz. *UCI Repository of machine learning databases.* University of California, Irvine, Dept. of Information and Computer Sciences, 1998.

[3] Olcay Boz. *Converting a trained neural network to a decision tree.* doctoral thesis, 2000.

[4] W. Duch and K. Grabczewski. *Searching for optimal MLP.* Fourth Conference on Neural Networks and Their Applications, Zakopane, May 1999.

[5] G.G.Towell and J.W.Shavlik. *Knowledge-based Artificial Neural Networks.* Artificial Intelligence Volume 69, 1992.

[6] M Sgarbi L.M Reyneri. *Performance of Weighted Radial Basis Function Classifiers.* http://polimage.polito.it/ lmr/articoli/esna.97.pdf, 1997.

[7] M.W.Craven. *Extracting comprehensible models from trained neural networks.* Machine Learning: Proceedings of the eleventh International Conference, San Franciso, 1996.

[8] Ross Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, ISBN 1558602380, January 1993.

[9] R.I.Damper R.A.Finan, A.T.Sapeluk. *Comparison of Multilayer and RBF Neural networks for Text-dependent Speaker Recognition.* Proceedings International Conference on Neural Networks, 1996.

[10] S-H. Choi & P.I Rockett. *Training of Neural Classifiers with Condensed Datasets.* IEEE Transactions on Systems, Man & Cybernetics B, publication pending.

[11] The Course Team. *T396 Study Notes.* The Open University, Walton Hall, Milton Keynes UK, 2002.

[12] Shi Zhong and Joydeep Ghosh. *Decision Boundary Focused Neural Network Classifier.* In Intelligent Engineering Systems Through Artificial Neural Networks (ANNIE), ASME Press, November, 2000.