Report

Assignment 3: Predicate Logic

Chiara Mocetti, Ebad Malik – P10

Exercise 1

`slice/4`, describing the *extraction* of a slice from a list:

```
slice([X|_],1,1,[X]).
slice([X|Xs],1,K,[X|Ys]) :- K > 1,
   K1 is K - 1, slice(Xs,1,K1,Ys).
slice([_|Xs],I,K,Ys) :- I > 1,
   I1 is I - 1, K1 is K - 1, slice(Xs,I1,K1,Ys).
```

Let us read this definition *declaratively*, that is in terms of *relations* that hold between arguments, as described by the three clauses:

1. The extraction of the slice made by elements from index 1 to index 1 of the list `[X|_]` is the list `[X]`.

2. *If*
    1. the upper bound/limit index of the slice is greater than 1.
    2. and we initialize K1 to K-1.
    3. and the extraction of the slice made by elements from index 1 to index K1 of the list `Xs` is `Ys`.

    Then the extraction of the slice made by elements from index 1 to index K of the list `[X|Xs]` is `[X|Ys]`.

3. *If*
    1. the lower bound/limit index of the slice is greater than 1.
    2. and we initialize I1 to I-1.
    3. and we initialize K1 to K-1.
    4. and the extraction of the slice made by elements from index I1 to index K1 of the list `Xs` is `Ys`.

    Then the extraction of the slice made by elements from index I to index K of the list `[_|Xs]` is `Ys`.

Let us read `slice/4` (shown above) *procedurally* for the query `?- slice([1, 2, 3 ,4],2,3,L2).`

1. Does the *first* clause apply?

    No, because `[1, 2, 3 ,4]` does not unify with `[X|_]`. or because 2 does not unify with 1?

2. Does the *second* clause apply?

No, because `2` does not unify with `1`.

3. Does the *third* clause apply?

Due to the way resolution works, we must introduce *fresh variables* when considering a clause. So, let us use `Xs'`, `I'`, `K'`, and `Ys'` for the variables `Xs`, `I`, `K` and `Ys` that appear in the clause head. The answer is: **Yes**, the second clause applies with the bindings `Xs'=[1, 2, 3 , 4]`, `I'=2`, `K'=3`, `Ys'=Ys`.

Q1: Is 2 > 1?

R1: Yes.

R and Q 2: I1 is 2+ -1=1.

R and Q3: K1 is 3 + -1=2.

Q4: `slice([2, 3, 4],1,2,Ys').`

1. Does the *first* clause apply?

No, because `[2,3,4]` does not unify with `[X|_]`.

2. Does the *second* clause apply?

Let us use `X'`, `Xs''`, `K''` and `Ys''`. **Yes**, the clause applies with `X'=2`, `Xs''=[3,4]`, `K''=2` and `Ys''= Ys''`.

Q4.1 Is 2 > 1?

R4.1: Yes.

Q and R of 4.2: K1 is 2+ -1.

Q4.2: `slice([3, 4],1,1, Ys'').`

1. Does the *first* clause apply?

Yes. Let us use `X''` to denote the single variable of the first clause at this step of the computation. So the first clause applies with `X''=3` and `Ys''=X''`.

R4: `Ys''=[2,3].`

`Ys=[2,3].`

Exercise 9.5

For 9.5, We were unable to figure out how to extract/upload a csv file to be compatible with the programming environment in Swish. We manage to finish the parts upto **9.5 (**Application to a real database).

Although we understand how knn is implemented in 9.5, the struggle was with the csv file. We were going to use Leave-One-Out Cross Validation Technique which:

1. Splits a dataset into a training set and a testing set, using all but one observation as part of the training set.
2. Build the model using only data from the training set.
3. Use the model to predict the response value of the one observation left out of the model and calculate the Accuary or Error.

4. Repeat the process n times.
5. Then calculate the test MSE to be the average of all of the test MSE's.

This is useful since it tends not to overestimate the test MSE compared to using a single test set & provides a much less biased measure of test MSE compared to using a single test set because we repeatedly fit a model to a dataset that contains n-1 observations.

On the other hand, It is time-consuming and computationally expensive.