

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

**CZ3005 Artificial Intelligence**

**Assignment 4: Implementing a Talking Box with Prolog**

Name: Loh Yi Xuan Renice

Matriculation No.: U1822247D

Group: TSP6

Table of contents

Introduction……………………………………………………...…………………...3

Implementation ……………………………………………………………………...3

Analysis of predicates used…………………………………………...…………...4

Executing prolog……………………………………………………………………..5

Prolog demo………………………………………………………...……………….6

Appendix A……………………………………………………...…………………...7

Introduction

In this assignment, we develop a Dialogue AI with Prolog. The Prolog system is regarded as a parent, conversing with their child who can only answer yes or no. The parent is able to intelligently ask related questions or a random question of another topic depending on the answer of the kid.

Implementation

To solve the problem, we break down the problem into three parts.

1. ***Storing user’s previous choice.*** Similar to the implementation in TalkingBox Demo, everytime a user gives an answer to a question, assert() will be called. In this case, if the answer is positive, assert(did(X)) will be called. Else, assert(didNot(X)) will be called.
2. ***Reading user’s previous choice.*** As most of the logic is based on set operation, we need to collect the previous asserts to a list.
3. ***Determine the phase of the program.*** There are 2 different types of phases during the program, namely, asking non-related questions and asking related follow up questions. There are 7 main questions regarding eating, playing, making new friends, learning, manners, doing sports and lastly, resting. The user’s answer will determine the phase the program will be in. Each main question has its own list of related questions and each time it is asked, it will be subtracted from the list.

Analysis of predicates used

1. **queryActivity(L)**

Ask about the activity L and add activity to either the list ‘did’ or ‘didNot’ based on the answer

1. **queryActivity([ ])**

Check if there are any more questions remaining. Else, end code.

1. **checkAnswer(Y)**

Check if activity Y is in the list did. If it is, execute answerYes. Else, execute answerNo.

1. **answerYes(Y)**

Get list L of follow up questions related to activity Y

1. **answerNo(0）**

Get list L of activities and query if the child has the remaining activities in List L.

1. **optionsActivity(L)**

Select a random question from list L

1. **askFollowUp(L)**

Ask follow up questions based on list L

1. **optionsFollowUp(Y, L)**

Find list of related follow up questions L based on previous follow up question Y using relatedFollowUp

1. **relatedFollowUp(Y, X)**

Find object X which is a member of the same list as object Y

1. **queryFollowUp(L)**

Remove all the asked questions from the list L

1. **checkRemaining([])**

Check if the list is empty. If it is, query about another activity.

1. **checkRemaining(R)**

If list is not empty, ask follow up question and add question to list ‘asked’

1. **related(Y , X)**

Returns object X which is a random member of the list corresponding to variable Y.

1. **random(Y)**

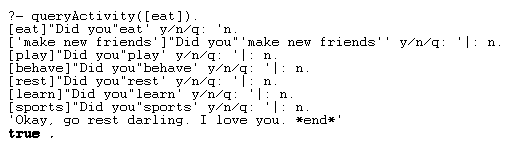
Returns random question from remaining objects from list Y

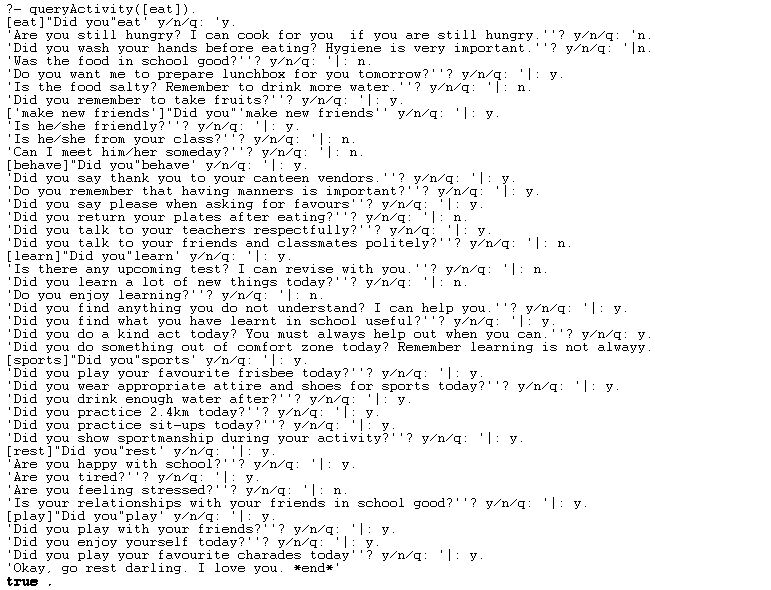
Executing prolog

To start the program, follow these steps:

1. In prolog, change working directory to the folder where talking\_to\_kid.pl file is located
2. Loading the talking\_to\_kid. file using [‘talking\_to\_kid.pl’]. command
3. Execute the predicate queryActivity([eat]).
4. Follow the prompt of the chatbox and reply ‘y’ and ‘n’ as necessary
5. When responding to the chat box, ‘y’ means yes, ‘n’ means no and ‘q’ means quit. Replies are in small letters.

Prolog Demo



**Screenshot 1: Kid did nothing** 

**Screenshot 2: Normal conversation**

**Appendix A**

**:- dynamic did/1.**

**:- dynamic didNot/1.**

**:- dynamic asked/1.**

**/\* Start query with first activity as eat \*/**

**ask(0):- queryActivity([eat]).**

**/\* Check if activity, Y, is in the list did. If yes, execute answerYes. If no execute answerNo \*/**

**checkAnswer(Y) :-**

**did(Y), answerYes(Y); answerNo(0).**

**/\* If selected activity not performed by child, get list L of activities. Ask question based on list L \*/**

**answerNo(0) :- optionsActivity(L), queryActivity(L).**

**/\* Select a random question from list L \*/**

**optionsActivity(L) :- findnsols(100,X,random(X),L).**

**/\* Check if there are no more activities to ask about. If yes, end code. \*/**

**queryActivity([]) :- print('Okay, go rest darling. I love you. \*end\*').**

**/\* Ask about activity, L, and add activity to either 'did' or 'didNot' based on answer \*/**

**queryActivity(L) :-**

**print(L), print("Did you"), member(X,L), print(X), print(' y/n/q: '), read(Answer), (Answer==q -> abort;Answer==y -> assert(did(X));assert(didNot(X))), checkAnswer(X).**

**/\* If chosen activity was done by child, get list of related follow up questions (L) corresponding to activity Y \*/**

**/\* Ask follow up question based on list L \*/**

**answerYes(Y) :- optionsFirstFollowUp(Y, L), queryFollowUp(L).**

**/\* Finds list L of related follow up questions corresponding to activty Y \*/**

**optionsFirstFollowUp(Y, L) :- findnsols(100, X, related(Y,X), L).**

**/\* Finds list of follow up questions, L, related to the previous follow up question Y \*/**

**/\* Ask follow up question based on list L \*/**

**askFollowUp(Y) :- optionsFollowUp(Y, L), queryFollowUp(L).**

**/\* Find list of related follow up questions, L, based on previous follow up question, Y, using relatedFollowUp\*/**

**optionsFollowUp(Y, L) :- findnsols(100, X, relatedFollowUp(Y,X), L).**

**/\* Finds all objects in list 'asked', convert the list to set\*/**

**/\* Remove objects in list asked from list L result is remaining. Checks if list is empty\*/**

**queryFollowUp(L) :-**

**findnsols(100,X,asked(X),Asked), list\_to\_set(L,S), list\_to\_set(Asked,A), subtract(S,A,Remaining), checkRemaining(Remaining).**

**/\* Checks input list is empty, by pattern match \*/**

**/\* If empty, no more follow up questions, ask about another activity \*/**

**checkRemaining([]) :- answerNo(0).**

**/\* If not empty, ask follow up question and add question to 'asked'\*/**

**checkRemaining(R) :- member(X,R), print(X), print('? y/n/q: '), read(Answer), (Answer==q -> abort;assert(asked(X))), askFollowUp(X).**

**/\* Finds rule to execute based on pattern match of first input variable \*/**

**/\* Returns object X which is a random member of the list corresponding to the first input variable. \*/**

**related(eat, X):- eat(L),random\_member(X, L).**

**related(play, X):- play(L),random\_member(X, L).**

**related('Make new friends', X):- friends(L),random\_member(X, L).**

**related(learn, X):- learn(L),random\_member(X, L).**

**related(behave, X):- behave(L),random\_member(X, L).**

**related('do sports', X):- sports(L),random\_member(X, L).**

**related(rest, X):- wellbeing(L),random\_member(X, L).**

**/\* Finds object X, which is a member of the same list as object Y \*/**

**relatedFollowUp(Y, X) :-**

**eat(L),member(X,L),member(Y,L);**

**play(L),member(X,L),member(Y,L);**

**sports(L),member(X,L),member(Y,L);**

**behave(L),member(X,L),member(Y,L);**

**wellbeing(L),member(X,L),member(Y,L);**

**learn(L),member(X,L),member(Y,L);**

**friends(L),member(X,L),member(Y,L).**

**/\* Removes already asked about activities from list activity. \*/**

**/\* Returns random activity from Remaining objects i.e from list Remaining \*/**

**random(Y) :- activity(A), findnsols(100,X,did(X),DidList), findnsols(100,X,didNot(X),DidNotList), append(DidList,DidNotList,History), list\_to\_set(A,S), list\_to\_set(History,H), subtract(S,H,Remaining), random\_member(Y, Remaining).**

**/\* List of activities \*/**

**activity([eat, play, 'make new friends', learn, behave, sports, rest]).**

**/\* Lists of follow up questions based on activity \*/**

**eat(['Did you wash your hands before eating? Hygiene is very important.', 'Was the food in school good?', 'Do you want me to prepare lunchbox for you tomorrow?', 'Is the food salty? Remember to drink more water.', 'Did you remember to take fruits?', 'Are you still hungry? I can cook for you if you are still hungry.']).**

**play(['Did you enjoy yourself today?', 'Did you play your favourite charades today', 'Did you play with your friends?']).**

**friends(['Is he/she from your class?', 'Is he/she friendly?', 'Can I meet him/her someday?']).**

**learn(['Did you learn a lot of new things today?', 'Do you enjoy learning?', 'Is there any upcoming test? I can revise with you.', 'Did you find anything you do not understand? I can help you.' , 'Did you find what you have learnt in school useful?', 'Did you do a kind act today? You must always help out when you can.', 'Did you do something out of your comfort zone today? Remember learning is not always about academics.']).**

**behave(['Do you remember that having manners is important?', 'Did you say thank you to your canteen vendors.', 'Did you say please when asking for favours' , 'Did you return your plates after eating?', 'Did you talk to your teachers respectfully?', 'Did you talk to your friends and classmates politely?']).**

**sports(['Did you wear appropriate attire and shoes for sports today?', 'Did you drink enough water after?', 'Did you play your favourite frisbee today?', 'Did you practice 2.4km today?', 'Did you practice sit-ups today?', 'Did you show sportsmanship during your activity?']).**

**wellbeing(['Are you tired?' , 'Are you feeling stressed?', 'Are you happy with school?', 'Is your relationships with your friends in school good?']).**

**/\* Initialize lists of activities done, not done \*/**

**did(nothing).**

**didNot(nothing).**

**/\* Initialise list of follow up questions asked \*/**

**asked(nothing).**