NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

CZ3005 : Artificial Intelligence lab report 2

DAI JUNWEI
N1902106B
TSR1

Kid’s day at school

Objective:

Assume that the prolog script is a parent, trying to know about a kid’s day at school. The prolog script should converse intelligently with the kid as follows. The prolog script should ask a question to the kid that kid can answer in yes or no. Depending on whether the answer is yes or no, prolog script should ask a related question or another random question. For example, if the kid says yes to whether it ate or not, the query can be a food item, or whether the kid used fork or spoon, or whether the kid washed hand. Similarly, related to games.

Overview:

The assignment has been completed with Prolog programming. The program does not have a GUI but the script interacts with the user by printing the questions on the screen in SWI- Prolog. After loading the prolog file in the memory, user can input ask(0) to start this conversation. The parents start by asking what did the kids do at school, did the child behave himself well. If the child says yes， the parents will then ask more detail like “did you say please”. The kids will reply yes or no. Then after finished all related question, parent may ask other question about the kid’s day at school. after asked all the questions. Parents will tell the kids to wash his hand and wait for dinner.

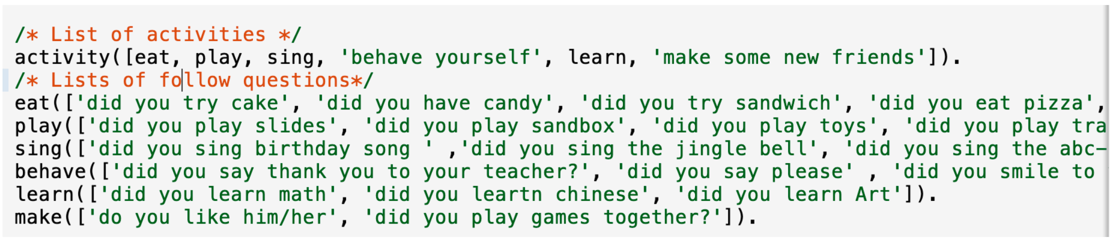
Rationale:

1. Create lists of different activities, like play, ear, sing, behave, weather.
2. Create rules for the relation between questions, which questions should the parents ask next.
3. response to the answer that the kids made and ask different question based upon it.
4. Already asked question will not show up again.

Explanation of the program:

Activity Lists:

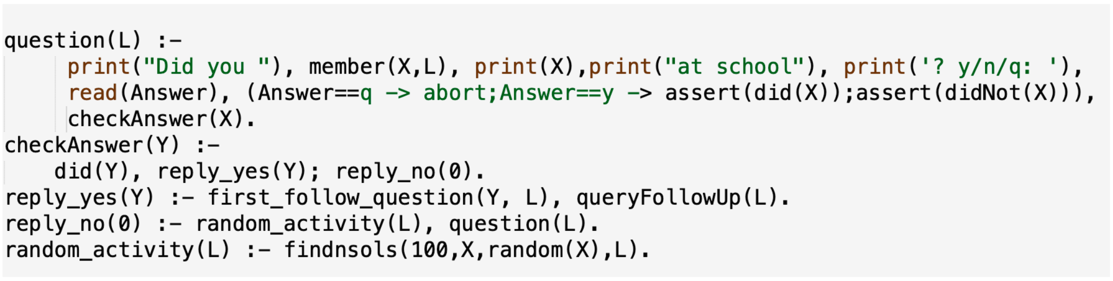
The activity lists are used to state the main topic that are about to ask, and the following question list are follow up questions based upon activity.



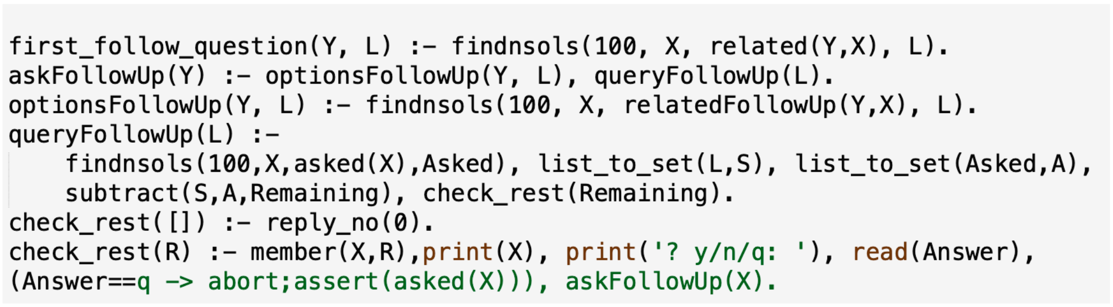
```
/* List of activities */
activity([eat, play, sing, 'behave yourself', learn, 'make some new friends']).
/* Lists of follow questions*/
eat(['did you try cake', 'did you have candy', 'did you try sandwich', 'did you eat pizza',
play(['did you play slides', 'did you play sandbox', 'did you play toys', 'did you play tra
sing(['did you sing birthday song ' ,'did you sing the jingle bell', 'did you sing the abc-
behave(['did you say thank you to your teacher?', 'did you say please' , 'did you smile to
learn(['did you learn math', 'did you leartn chinese', 'did you learn Art']).
make(['do you like him/her', 'did you play games together?']).
```

Question ask Rules:

Parents will ask the first question, and after asked this activity, based upon the kids' answer. It will be put into did list or did not list. If the kids did activity(L) in school, then parents will ask the following question in the activity(L). For example, if it is in the activity eat, parents will then ask did you eat Chinese food? Also, asked question will be subtracted from the original list.



```
question(L) :-
     print("Did you "), member(X,L), print(X),print("at school"), print('? y/n/q: '),
     read(Answer), (Answer==q -> abort;Answer==y -> assert(did(X));assert(didNot(X))),
     checkAnswer(X).
checkAnswer(Y) :-
    did(Y), reply_yes(Y); reply_no(0).
reply_yes(Y) :- first_follow_question(Y, L), queryFollowUp(L).
reply_no(0) :- random_activity(L), question(L).
random_activity(L) :- findnsols(100,X,random(X),L).
```



```
first_follow_question(Y, L) :- findnsols(100, X, related(Y,X), L).
askFollowUp(Y) :- optionsFollowUp(Y, L), queryFollowUp(L).
optionsFollowUp(Y, L) :- findnsols(100, X, relatedFollowUp(Y,X), L).
queryFollowUp(L) :-
    findnsols(100,X,asked(X),Asked), list_to_set(L,S), list_to_set(Asked,A),
    subtract(S,A,Remaining), check_rest(Remaining).
check_rest([]) :- reply_no(0).
check_rest(R) :- member(X,R),print(X), print('? y/n/q: '), read(Answer),
(Answer==q -> abort;assert(asked(X))), askFollowUp(X).
```

Let us look one by one at the clauses in the rule to ask questions to the kids.

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

First, the question() gets a list as a parameter. The list is the activity list. Then the read() reads the y or n from the user and assert it in did() list or didNot() list. Use checkAnswer() to find out what to do next.

```
checkAnswer(Y) :-
    did(Y), reply_yes(Y); reply_no(0).
reply_yes(Y) :- first_follow_question(Y, L), queryFollowUp(L).
reply_no(0) :- random_activity(L), question(L).
random_activity(L) :- findnsols(100,X,random(X),L).
```

Check if L is in the did list, if it is then execute reply_yes(), otherwise, execute reply_no(). If the answer is yes,then, it will ask the first follow up question that is in the activity L,

```
queryFollowUp(L) :-
    findnsols(100,X,asked(X),Asked), list_to_set(L,S), list_to_set(Asked,A),
    subtract(S,A,Remaining), check_rest(Remaining).
check_rest([]) :- reply_no(0).
check_rest(R) :- member(X,R),print(X), print('? y/n/q: '), read(Answer),
(Answer==q -> abort;assert(asked(X))), askFollowUp(X).
```

Finds all objects in list 'asked', convert the list to set, remove objects in list asked from list\object L result is Remainging. Checks if Remaining is empty, If empty, no more follow up questions, ask about another activity. If not empty, ask follow up question and add question to 'asked'.

Rules to find related questions:

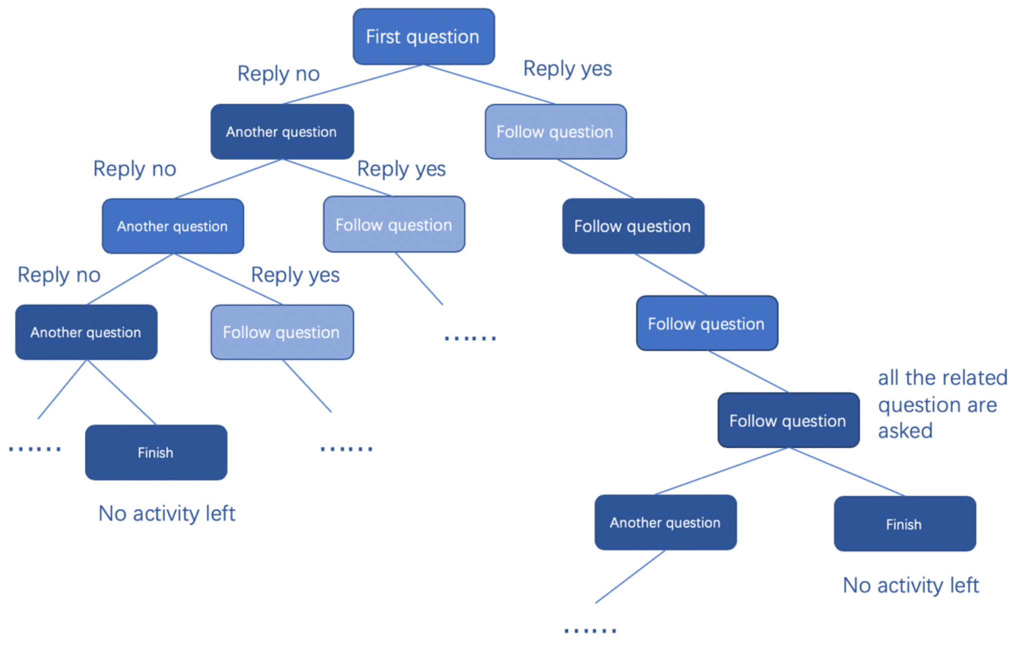
Related() is used to find the first follow question of activity, it connect the activity to the follow list. For example, if the activity is eat , the program will choose a random member in the eat list and ask question about it.

```
related(eat, X):- eat(L),random_member(X, L).
related(play, X):- play(L),random_member(X, L).
related(sing, X):- sing(L),random_member(X, L).
related('behave yourself', X):- behave(L),random_member(X, L).
related(learn, X):- learn(L),random_member(X, L).
related('make some new friends', X):- make(L),random_member(X, L).
```

```
relatedFollowUp(Y, X) :-
    eat(L),member(X,L),member(Y,L);
    play(L),member(X,L),member(Y,L);
    sing(L),member(X,L),member(Y,L);
    behave(L),member(X,L),member(Y,L);
    learn(L),member(X,L),member(Y,L);
    make(L),member(X,L),member(Y,L).
```

The relatedFollowUp() is used to find object X, which is a member of the same list as object Y.

Logic Tree:



Appendix A:

Prolog Script:

```
:-dynamic(did/1).
:-dynamic(didNot/1).
:-dynamic(asked/1).


/* Get start with 'ask(0)',nl is for jump into next line */
ask(0):- print('Welcome home,sweetie,what did you do today?'),nl,question(['
behave yourself']).
/* Check if is empty. If yes, end code. */
```

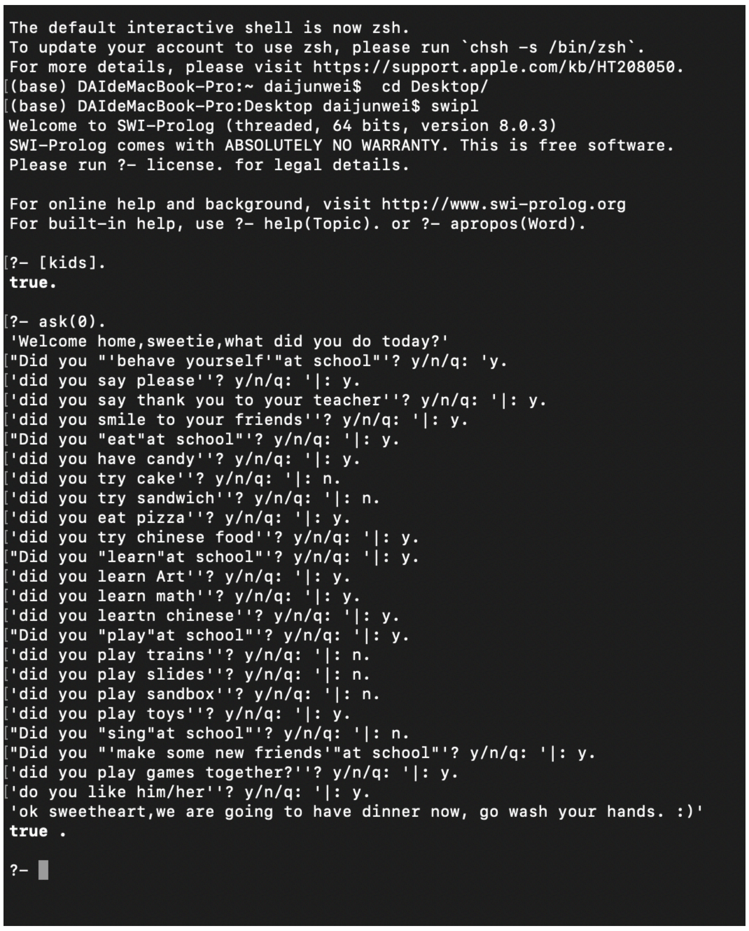
```
question([]) :- print('ok sweetheart,we are going to have dinner now, go wash your hands. :)').

/* Ask question of different activities,  add activity to "did" or "didNot" */
question(L) :-
     print("Did you "), member(X,L), print(X),print("at school"), print('? y/n/q: '), read(Answer), (Answer==q -> abort;Answer==y -> assert(did(X));assert(didNot(X))), checkAnswer(X).
/* after "question(L)",the reply of child is either y or n. check the answer,If yes, execute execute"reply_yes". If no execute "reply_no" */
checkAnswer(Y) :-
    did(Y), reply_yes(Y); reply_no(0).

/*if the child reply yes , we need to ask follow question that is in the follow list L*/
/*L is related to activity Y*/
reply_yes(Y) :- first_follow_question(Y, L), queryFollowUp(L).


/* if the child replys no,we need to ask whether the child did other activities. Ask question based upon list L */
reply_no(0) :- random_activity(L), question(L).

random_activity(L) :- findnsols(100,X,random(X),L).
/*when kids says yes, go into the list of activity Y.and ask question in it. */
first_follow_question(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 questin based upon list L */
askFollowUp(Y) :- optionsFollowUp(Y, L), queryFollowUp(L).
/* Find list of related follow up questions, L, based upon 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\object L result is Remainging. Checks if Remaining is empty*/
queryFollowUp(L) :-
    findnsols(100,X,asked(X),Asked), list_to_set(L,S), list_to_set(Asked,A), subtract(S,A,Remaining), check_rest(Remaining).
```

```prolog



/* If empty, no more follow up questions, ask about another activity */
check_rest([]) :- reply_no(0).
/* If not empty, ask follow up question and add question to 'asked'*/
check_rest(R) :- member(X,R),print(X), print('? y/n/q: '), read(Answer), (Answer==q -> abort;assert(asked(X))), askFollowUp(X).


/*when the kids says yes about "eat", it will randomly find follow question in the eat list. */
/*it means random_member X are related to the activity */
related(eat, X):- eat(L),random_member(X, L).
related(play, X):- play(L),random_member(X, L).
related(sing, X):- sing(L),random_member(X, L).
related('behave yourself', X):- behave(L),random_member(X, L).
related(learn, X):- learn(L),random_member(X, L).
related('make some new friends', X):- make(L),random_member(X, L).



/*  X and Y are all in the same list L ,so parents will ask the related question. */
relatedFollowUp(Y, X) :-
    eat(L),member(X,L),member(Y,L);
    play(L),member(X,L),member(Y,L);
    sing(L),member(X,L),member(Y,L);
    behave(L),member(X,L),member(Y,L);
    learn(L),member(X,L),member(Y,L);
    make(L),member(X,L),member(Y,L).



/* subtract already asked question from list */
/* 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, sing, 'behave yourself', learn, 'make some new friends']).
```

```
/* Lists of follow questions*/
eat(['did you try cake', 'did you have candy', 'did you try sandwich', 'did you eat pizza','did you try chinese food']).
play(['did you play slides', 'did you play sandbox', 'did you play toys', 'did you play trains']).
sing(['did you sing birthday song ' ,'did you sing the jingle bell', 'did you sing the abc-song']).
behave(['did you say thank you to your teacher', 'did you say please' , 'did you smile to your friends']).
learn(['did you learn math', 'did you leartn chinese', 'did you learn Art']).
make(['do you like him/her', 'did you play games together?']).

did(nothing).
didNot(nothing).
asked(nothing).
```

Appendix B:

Sample Run: Kid's day at school



```
The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
[(base) DAIdeMacBook-Pro:~ daijunwei$  cd Desktop/
[(base) DAIdeMacBook-Pro:Desktop daijunwei$ swipl
Welcome to SWI-Prolog (threaded, 64 bits, version 8.0.3)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit http://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

[?- [kids].
true.

[?- ask(0).
'Welcome home,sweetie,what did you do today?'
["Did you "'behave yourself'"at school"'? y/n/q: 'y.
['did you say please''? y/n/q: '|: y.
['did you say thank you to your teacher''? y/n/q: '|: y.
['did you smile to your friends''? y/n/q: '|: y.
["Did you "eat"at school"'? y/n/q: '|: y.
['did you have candy''? y/n/q: '|: y.
['did you try cake''? y/n/q: '|: n.
['did you try sandwich''? y/n/q: '|: n.
['did you eat pizza''? y/n/q: '|: y.
['did you try chinese food''? y/n/q: '|: y.
["Did you "learn"at school"'? y/n/q: '|: y.
['did you learn Art''? y/n/q: '|: y.
['did you learn math''? y/n/q: '|: y.
['did you leartn chinese''? y/n/q: '|: y.
["Did you "play"at school"'? y/n/q: '|: y.
['did you play trains''? y/n/q: '|: n.
['did you play slides''? y/n/q: '|: n.
['did you play sandbox''? y/n/q: '|: n.
['did you play toys''? y/n/q: '|: y.
["Did you "sing"at school"'? y/n/q: '|: n.
["Did you "'make some new friends'"at school"'? y/n/q: '|: y.
['did you play games together?''? y/n/q: '|: y.
['do you like him/her''? y/n/q: '|: y.
'ok sweetheart,we are going to have dinner now, go wash your hands. :)'
true .

?- 
```