NANYANG TECHNOLOGICAL UNIVERSITY

CZ3005
ARTIFICIAL INTELLIGENCE

Assignment 4 Report

Talking Box with Prolog

Qn3: Subway Sandwich Interactor

Written by: MOCK JIN WEI

Tutorial Group: TSP1

Matriculation Number: 1721242F

Email: JMOCK001@e.ntu.edu.sg

1. Scenario

For this assignment, question 3: Subway Sandwich Interactor is chosen.

The following scenario is considered for the implementation of the Subway Sandwich Interactor Prolog program:

The interactor will allows user to select 4 different meal choices and they are: 1. Vegan, 2. Healthy, 3. Veggie and 4. Value meals. Depending on the user's meal selection, the program will intelligently display the corresponding ingredients categories for the user to choose from. For example, meat category will not be shown to the user if the user has selected a Vegan meal. In addition, the program can also filter out ingredients that does not fit in the selected meal, even though the category is available for user to choose from. An example is that while Healthy meal offers sides, it will not offer chips as an option for the user.

Table 1 below highlights the different available ingredients category for each of the meals.

Vegan Meal	**Healthy Meal**
Available ingredients category: 1. Bread 2. Veggie 3. Condiment 4. Drink 5. Sides *Ingredients for Sides will be filtered accordingly to display the relevant choices for Vegan Meal.	Available ingredients category: 1. Bread 2. Meat 3. Veggie 4. Topping 5. Condiment 6. Drink 7. Sides *Ingredients for all category will be filtered accordingly to display the relevant choices for Healthy Meal.
Veggie Meal	**Value Meal**
Available ingredients options: 1. Bread 2. Veggie 3. Condiment 4. Topping 5. Drink 6. Sides *Ingredients for Sides will be filtered accordingly to display the relevant choices for Vegan Meal.	Available ingredients options: 1. Bread 2. Meat 3. Veggie 4. Condiment 5. Drink

**Table 1*

**Ingredients options are referenced from the actual Subway' menu, with some modification made to better suit the Prolog program.*

**For more details on what are the specific breads, veggies, meats, etc. being offered for each category and for each meal choices, please refer to A1 and A2 in the appendix.*

A normal flow of the Subway Sandwich Interactor program will starts from the user selecting a meal option, followed by choosing the ingredients from the corresponding category. At the end of the program, the interactor will display all the selected ingredients that the user has chosen. For example, if the user had chosen the "Value" meal, the flow will be as flows:

Select Value meal → Choose Bread → Choose Meats → Choose Veggie → Choose Condiments → Choose Drink → Subway Sandwich Interactor ends the selection and displays all the selected ingredients.

2. Prolog Program Design

This section will explain the design behind the implemented Subway Sandwich Interactor.

Before user can use the Subway Sandwich Interactor program, the user will have to first run the *dynamic(X)*, where X is the function names that are found in Fig 1. By running the *dynamic(X)* code, it will allows the Prolog program to dynamically update the KD with the knowledge gained from each user's selection in the program. These functions will acts as a temporary "database" that holds all user's selections.

```
% Each ingredient is "initialized" with an empty assertion
selected_meat(nothing).
selected_veg(nothing).
selected_topping(nothing).
selected_condiment(nothing).
selected_drink(nothing).
selected_sides(nothing).
selected_bread(nothing).
```

**Fig 1. Function names for dynamic().*

After running the *dynamic()* code, the program will be ready for the user to use. To start the Subway Sandwich Interactor program, user will just have to enter the "ask(0)" into the program. This will result in the program displaying the list of available meals option for the user to choose from. As seen from Fig 2 below, the program is designed to provide user with 4 different meal choices.

```
ask(0):-
    write(" "), nl,
    write("Welcome to Subway! What type of meal do you want to have?"), nl,
    write("Please the number of your meal choice:"), nl,
    write("*******************************"), nl,
    write("1: Vegan "), nl,
    write("2: Healthy "), nl,
    write("3: Veggie "), nl,
    write("4: Value "), nl,
    write("*******************************"), nl,

    %Allows the program to take in the user's choice.
    read(Selected),

    %Depending on the selected meal choice, the corresponding meal category ingredients will be displayed.
    assertz(meal_selected(Selected)),
    meal_options(0).
```

**Fig 2. Meals Selection.*

With references to Fig 2:
read → Take in the user's choices for meal

assertz → Stores the selected meal choice and will be used to determine the corresponding meal categories ingredients that is associated with the selected meal.

Depending on the user's meal selection, different categories of ingredients will be prompted for the user to choose from. With references to Table 1, the filtering of category ingredients for the different meal options are done by the following codes:

```
%Display the list of ingredients corresponding to Vegan.
meal_selected(1) ->
bread(BreadList), veg(VegList), condiment(CondimentList), drink(DrinkList), sides(SidesList), vegan(VeganList),

%findall helps to filter out the available ingredients for Vegan Meal.
findall(A, ( member(A, BreadList), member(A, VeganList) ), FilteredBread ),
findall(C, ( member(C, VegList), member(C, VeganList) ), FilteredVeg ),
findall(E, ( member(E, CondimentList), member(E, VeganList) ), FilteredCondiment ),
findall(F, ( member(F, DrinkList), member(F, VeganList) ), FilteredDrink ),
findall(G, ( member(G, SidesList), member(G, VeganList) ), FilteredSides ),
```

**Fig 3. Ingredients Selection for Vegan Meal.*

```
%Display the list of ingredients corresponding to Healthy.
meal_selected(2) ->
bread(BreadList), veg(VegList), meat(MeatList), topping(ToppingList), drink(DrinkList), condiment(CondimentList),
sides(SidesList), healthy(HealthyList),

%findall helps to filter out the available ingredients for Vegan Meal.
findall(A, ( member(A, BreadList), member(A, HealthyList) ), FilteredBread ),
findall(B, ( member(B, MeatList), member(B, HealthyList) ), FilteredMeat),
findall(C, ( member(C, VegList), member(C, HealthyList) ), FilteredVeg ),
findall(D, ( member(D, ToppingList), member(D, HealthyList) ), FilteredTopping ),
findall(E, ( member(E, CondimentList), member(E, HealthyList) ), FilteredCondiment ),
findall(F, ( member(F, DrinkList), member(F, HealthyList) ), FilteredDrink ),
findall(G, ( member(G, SidesList), member(G, HealthyList) ), FilteredSides ),
```

**Fig 4. Ingredients Selection for Healthy Meal.*

```
%Display the list of ingredients corresponding to Veggie.
meal_selected(3) ->
bread(BreadList), veg(VegList), topping(ToppingList), condiment(CondimentList), drink(DrinkList), sides(SidesList),
veggie(VeggieList),

%findall helps to filter out the available ingredients for Veggie Meal.
findall(A, ( member(A, BreadList), member(A, VeggieList) ), FilteredBread ),
findall(C, ( member(C, VegList), member(C, VeggieList) ), FilteredVeg ),
findall(D, ( member(D, ToppingList), member(D, VeggieList) ), FilteredTopping ),
findall(E, ( member(E, CondimentList), member(E, VeggieList) ), FilteredCondiment ),
findall(F, ( member(F, DrinkList), member(F, VeggieList) ), FilteredDrink ),
findall(G, ( member(G, SidesList), member(G, VeggieList) ), FilteredSides ),
```

**Fig 5. Ingredients Selection for Veggie Meal.*

```
%Display the list of ingredients corresponding to Value.
meal_selected(4) ->
bread(BreadList), veg(VegList), meat(MeatList), condiment(CondimentList), drink(DrinkList),
value(ValueList),

%findall helps to filter out the available ingredients for Value Meal.
findall(A, ( member(A, BreadList), member(A, ValueList) ), FilteredBread ),
findall(B, ( member(B, MeatList), member(B, ValueList) ), FilteredMeat),
findall(C, ( member(C, VegList), member(C, ValueList) ), FilteredVeg ),
findall(E, ( member(E, CondimentList), member(E, ValueList) ), FilteredCondiment ),
findall(F, ( member(F, DrinkList), member(F, ValueList) ), FilteredDrink ),
```

**Fig 6. Ingredients Selection for Value Meal.*

As seen from Fig 3 to Fig 6, the way to filter the corresponding ingredients for the selected meal involves the usage of *"meal_selected"* and *"findall"*.

meal_selected(X) → Specify the relevant ingredients categories that is associated with the selected meal. All ingredients in the list (E.g. BreadList, VegList) are retrieved from the "database" that is shown in A1 and A2 in the appendix. X here is the meal's number.

findall(A…….) → To further filter out the ingredients in a particular category. Through the keyword in "*findall"* that specify the current selected choice (E.g. VeganList), the program will be able to find all available ingredients for the selected meal and will compare it and find the intersection ingredients in both the *"meal_selected"* and "findall" function.

After the program had completed finding the intersecting ingredients for the selected choice of meal, it will display the list of ingredient, grouped by category. One category will only appear at any single time and only after the user had selected the ingredient from a category will then the next list of ingredient from another category will appear. For example, the user will first have to choose a type of bread, before they are allowed to choose the next ingredient option (Could be Meat or Veggie depending on the user's chosen meal).

The code snippet below will shows how the Subway Sandwich Interactor display categories and lists of ingredients for user to choose. As similar set of codes will be used for each of the category, only codes for the Bread and Meat category it will be shown here. The rest of the categories (E.g. Toppings, Drinks), will have similar codes to display their ingredients.

```
/* Bread Selection */
write(" "), nl,
write("*********************************"), nl,
write("Please select a bread: "), nl,
write("*********************************"), nl,

print_options(FilteredBread), nl,
read(SelectedBread),
assertz(selected_bread(SelectedBread)),

/* Meat Selection */
write(" "), nl,
write("*********************************"), nl,
write("Please select a meat: "), nl,
write("*********************************"), nl,

print_options(FilteredMeat), nl,
read(SelectedMeat),
assertz(selected_meat(SelectedMeat)),
```

**Fig 7. Ingredients Selection for a Bread and Meat Selection*

Based on Fig 7:
**XX here means the category type (E.g. Bread, Meat, Veggie, Topping, Drinks, Condiments, and Sides).*

print_option(FilteredXX) → Print out the list of ingredients that are in XX category and is applicable to the user's selected meal.

read(SelectedXX) → Similar to "*read"* that is seen from Fig 2. Read user's choice of ingredient.

assertz(selected_XX(SelectedXX)) → Stores the selected XX option into the corresponding dynamic function that is seen in Fig 1.

---Continue on next page---

Once Subway Sandwich Interactor has finished asking the user for their selection, the program will read all store “data” in the dynamic function that is seen from Fig 1 and display it on screen for the user to view. The displayed “data” will undergo some form of formatting as seen from the Fig 8 below before it gets displayed.

```
done(1) is a function that prints out all the selections that the users
has made.
*/
done(1):-
    write(" "), nl,
    write("*********************************"), nl,
    write("Enjoy your meal!"), nl,
    write("*********************************"), nl,

    (\+ ( selected_bread(X), bread(Bread), member(X, Bread) ) -> write("No bread selected."), nl;
    ( write("Selected bread: "), selected_bread(X), bread(Bread), member(X, Bread), write(X), nl )),

    (\+ ( selected_meat(Y), meat(Meat), member(Y, Meat) ) -> write("No meat selected."), nl;
    ( write("Selected meat: "), selected_meat(Y), meat(Meat), member(Y, Meat), write(Y), nl )),

    (\+ selected_veg(_) -> write("No veggie selected."), nl;
    ( veg(VegList), findall(Z,(selected_veg(Z), member(Z, VegList)), List), write("Selected veggie: "), nl ,print_options(Li
st) )),

    (\+ ( selected_topping(A), topping(Topping), member(A, Topping)) -> write("No topping selected."), nl;
    ( write("Selected topping: "), selected_topping(A),topping(Topping), member(A,Topping), write(A), nl )),

    (\+ ( selected_condiment(B), condiment(Condiment), member(B, Condiment) ) -> write("No condiment selected."), nl;
    ( write("Selected condiment: "), selected_condiment(B), condiment(Condiment), member(B, Condiment), write(B), nl )),

    (\+ ( selected_drink(C), drink(Drink), member(C, Drink) ) -> write("No drink selected."), nl;
    ( write("Selected drink: "), selected_drink(C), drink(Drink), member(C, Drink), write(C), nl )),

    (\+ ( selected_sides(D), sides(Sides), member(D, Sides) ) -> write("No sides selected."), nl;
    ( write("Selected sides: "), selected_sides(D), sides(Sides), member(D, Sides), write(D), nl )),
```

**Fig 8. Prints out all user selections*

***Do note that the formatting for select_veg is different from others as the original plan for veggies selection has some errors. The display format for the veggies still remains the same as the original plan to avoid any unintentional display errors that might happen should changes be made to it.*

After displaying all the user’s selections and having a successful interactions with the user, the Subway Sandwich Interactor will reset all “data” selections and prepare the program for the next run. This is done by the following code:

```
/*
retractall is used to reset the program so that the Subway Sandwich Interactor will be a fresh clean program for the next ru
n.
*/
    retractall(selected_veg(_)),
    retractall(selected_bread(_)),
    retractall(selected_meat(_)),
    retractall(selected_topping(_)),
    retractall(selected_drink(_)),
    retractall(selected_sides(_)),
    retractall(selected_condiment(_)),
    retractall(meal_selected(_)),
```

**Fig 9. Resetting of the dynamic functions*

The retractall function seen above will be used to reset all “data” selection in the respective dynamic function. By the end of this execution, the corresponding function (E.g. selected_veg) will be empty and is ready to take in new selections for the next run.

---Continue on next page---

3. Interaction with program

This section will shows screenshots of the interaction made with the Subway Sandwich Interactor.

All interactions will begin with the following commands:

```
?- dynamic(selected_bread/1).
true.

?- dynamic(selected_meat/1).
true.

?- dynamic(selected_veg/1).
true.

?- dynamic(selected_topping/1).
true.

?- dynamic(selected_condiment/1)
true.

?- dynamic(selected_drink/1).
true.

?- dynamic(selected_sides/1).
true.
```

Interaction 1:

Ordering a Healthy Meal. A flatbread with ham, green peppers and topped up with cornflakes. The bread comes with mustard sauce. The meal also consists of orange juice and a cup of yogurt.

```
?- ask(0).

Welcome to Subway! What type of meal do you want to have?
Please the number of your meal choice:
*********************************
1: Vegan
2: Healthy
3: Veggie
4: Value
*********************************
|: 2.

*********************************
Please select a type of bread:
*********************************
italian
multigrain
flatbread
wrap

|: mulitgrain.

*********************************
Please select a meat:
*********************************
teriyakichicken
ham
roastbeef

|: ham.

*********************************
Please select a veggie:
*********************************
avocado
cucumbers
lettuce
tomatoes
onions
greenpeppers
olives
pickles

|: greenpeppers.
```

```
*********************************
Please select a topping:
*********************************
mushroomslices
cornflakes

|: cornflakes.

*********************************
Please select a condiment:
*********************************
mustard
sweetonion

|: mustard.

*********************************
Please select a side:
*********************************
fruitbar
yogurt

|: yogurt.

*********************************
Please select a drink:
*********************************
orangejuice
greentea

|: orangejuice.

*********************************
Enjoy your meal!
*********************************
Selected bread: flatbread
Selected meat: ham
Selected veggie:
greenpeppers
Selected topping: cornflakes
Selected condiment: mustard
Selected drink: orangejuice
Selected sides: yogurt
% Execution Aborted
```

---Continue on next page---

Interaction 2:

Ordering a Value Meal. An Italian bread with meatballs and cucumbers. The bread comes with ranch sauce. The meal also consists of root beer.

```
?- ask(0).

Welcome to Subway! What type of meal do you want to have?
Please the number of your meal choice:
*******************************
1: Vegan
2: Healthy
3: Veggie
4: Value
*******************************
|: 4.

*******************************
Please select a bread:
*******************************
italian
multigrain
honeyoat
flatbread
wrap

|: italian.

*******************************
Please select a meat:
*******************************
whitechicken
teriyakichicken
bologna
salami
pepperoni
ham
meatballs
roastbeef
bacon
tuna
egg
veggiepatty

|: meatballs.

*******************************
Please select a veggie:
*******************************
avocado
cucumbers
lettuce
tomatoes
onions
greenpeppers
olives
pickles

|: cucumbers.

*******************************
Please select a condiment:
*******************************
chipotle
mayonnaise
mustard
oliveoil
ranch
sweetonion
bbq
chilli
redwinevinegar
ketchup

|: ranch.

*******************************
Please select a drink:
*******************************
rootbeer
cocacola
mountaindew
orangejuice
greentea
coffee

|: rootbeer.

*********************************
Enjoy your meal!
*********************************
Selected bread: italian
Selected meat: meatballs
Selected veggie:
cucumbers
No topping selected.
Selected condiment: ranch
Selected drink: rootbeer
No sides selected.
% Execution Aborted
?-
```

---Continue on next page---

Interaction 3:

Ordering a Vegan Meal. A warp bread comes with pickles and bbq sauce. The meal also consists of a coca-cola and a fruit bar.

```
?- dynamic(selected_veg/1).
true.

?- dynamic(selected_topping/1).
true.

?- dynamic(selected_meat/1).
true.

?- dynamic(selected_sides/1).
true.

?- dynamic(selected_drink/1).
true.

?- dynamic(selected_bread/1).
true.

?- dynamic(selected_condiment/1).
true.

?- ask(0).

Welcome to Subway! What type of meal do you want to have?
Please the number of your meal choice:
*******************************
1: Vegan
2: Healthy
3: Veggie
4: Value
*******************************
|: 1.

*******************************
Please select a type of bread:
*******************************
italian
multigrain
honeyoat
flatbread
wrap

|: mulitgrain.
```

```
*******************************
Please select a veggie:
*******************************
avocado
cucumbers
lettuce
tomatoes
onions
greenpeppers
olives
pickles

|: pickles.

*******************************
Please select a condiment:
*******************************
mustard
oliveoil
sweetonion
bbq
chilli
redwinevinegar
ketchup

|: bbq.

*******************************
Please select a drink:
*******************************
rootbeer
cocacola
mountaindew
orangejuice
greentea
coffee

|: cocacola.
```

```
*********************************
Please select a side:
*********************************
chips
hashbrown
fruitbar
yogurt

|: fruitbar.

*********************************
Enjoy your meal!
*********************************
Selected bread: wrap
No meat selected.
Selected veggie:
pickles
No topping selected.
Selected condiment: bbq
Selected drink: cocacola
Selected sides: fruitbar
% Execution Aborted
?-
```

**Kindly ignore the "dynamic()" commands as it was screenshotted accidently.*

---Continue on next page---

Interaction 4:

Ordering a Veggie Meal. A honey oat bread that comes with onions, topped up with mushroom slices. The bread also come with red wine vinegar condiment. The meal is served with root beer and a fruit bar.

```
?- ask(0).

Welcome to Subway! What type of meal do you want to have?
Please the number of your meal choice:
*******************************
1: Vegan
2: Healthy
3: Veggie
4: Value
*******************************
|: 3.

*******************************
Please select a bread:
*******************************
italian
multigrain
honeyoat
flatbread
wrap

|: honeyoat.

*******************************
Please select a veggie:
*******************************
avocado
cucumbers
lettuce
tomatoes
onions
greenpeppers
olives
pickles

|: onions.
```

```
*******************************
Please select a topping:
*******************************
mushroomslices
cornflakes
americancheese
montereycheddar

|: mushroomslices.

*******************************
Please select a condiment:
*******************************
chipotle
mayonnaise
mustard
oliveoil
ranch
sweetonion
bbq
chilli
redwinevinegar
ketchup

|: redwinevinegar.

*******************************
Please select a side:
*******************************
cookie
chips
hashbrown
fruitbar
yogurt
|: fruitbar.
```

```
*********************************
Please select a drink:
*********************************
rootbeer
mountaindew
orangejuice
greentea
coffee

|: rootbeer.

*********************************
Enjoy your meal!
*********************************
Selected bread: honeyoat
No meat selected.
Selected veggie:
onions
Selected topping: mushroomslices
Selected condiment: redwinevinegar
Selected drink: rootbeer
Selected sides: fruitbar
% Execution Aborted
```

---End of Assignment 4 Report---

Appendix

```
%Category Ingredients
bread([italian, multigrain, honeyoat, flatbread, wrap]).

meat([whitechicken, teriyakichicken, bologna, salami, pepperoni, ham, meatballs, roastbeef, bacon, tuna, egg, veggiepatty]).

veg([avocado, cucumbers, lettuce, tomatoes, onions, greenpeppers, olives, pickles]).

topping([mushroomslices, cornflakes, americancheese, montereycheddar]).

condiment([chipotle, mayonnaise, mustard, oliveoil, ranch, sweetonion, bbq, chilli, redwinevinegar, ketchup]).

drink([rootbeer, cocacola, mountaindew, orangejuice, greentea, coffee]).

sides([cookie,chips, hashbrown , fruitbar, yogurt]).
```

A1: List of ingredients that each category has.

```
%Meals Ingredients
vegan([italian, multigrain, honeyoat, flatbread, wrap, avocado, cucumbers, lettuce, tomatoes, onions, greenpeppers, olives,
pickles, bbq, chilli, mustard, oliveoil, redwinevinegar, sweetonion, ketchup, chips, hashbrown , fruitbar, yogurt , rootbeer
, cocacola, mountaindew , orangejuice, greentea, coffee]).
healthy([italian, multigrain, flatbread, wrap, avocado, cucumbers, lettuce, tomatoes, onions, greenpeppers, olives, pickles,
 mushroomslices, cornflakes, teriyakichicken, ham, roastbeef, mustard, sweetonion, fruitbar, yogurt, orangejuice, greentea])
.
veggie([italian, multigrain, honeyoat, flatbread, wrap, veggiepatty, chipotle, honeymustard,avocado, cucumbers, lettuce, tom
atoes, onions, greenpeppers, olives, pickles, mushroomslices, cornflakes, americancheese, montereycheddar, mayonnaise, musta
rd, oliveoil, ranch, sweetonion, bbq, chilli, redwinevinegar, ketchup, cookie ,chips, hashbrown , fruitbar, yogurt,rootbeer,
cocacola ,mountaindew, orangejuice, greentea, coffee]).
value([italian, multigrain, honeyoat, flatbread, wrap, whitechicken, teriyakichicken, bologna, salami, pepperoni, ham, meatb
alls, roastbeef,bacon, tuna, egg, veggiepatty, avocado, cucumbers, lettuce, tomatoes, onions, greenpeppers, olives, pickles,
 chipotle, mayonnaise, mustard, oliveoil, ranch, sweetonion, bbq, chilli, redwinevinegar, ketchup, rootbeer,cocacola, mounta
indew, orangejuice, greentea, coffee]).
```

A1: List of ingredients that each meal option has.

*To avoid any typo and spacing errors, all ingredients name do not have space in between them.