Assignment Artificial Intelligence CZ3005

Subway sandwich interactor

Koch Philipp Frederik Edward, N1903454H

2019
November

1 Task

The goal of the task was the implementation of a subway sandwich interactor. This was to guide the customer through the selection. It had to be taken into account that some options could be chosen more often, such as the chosen vegetable. Furthermore, a restriction must be made by previously selected options, so that no meat-containing ingredients are permitted in the vegetarian menu. Assistance was offered for the task, which was also used. An options/1 rule and a selected/2 rule were proposed in the notes. The options/1 rule should offer the possible options for the respective sandwich part and the selected/2 rule should assign an option for the respective sandwich part. selected(0) should trigger a jump on the list of sandwich parts and initiate the next assignment. If X = 1, a done/1 rule should display the options already selected.

2 Implementation

The hints described above are very suitable for a command line based dialog program and were therefore chosen as the basis for the following implementation. For this type of program, where a rule is called again each time, an abstracted

state is indispensable. The selected properties of the sandwich, the state of the selection and the number of possible reusable options must be saved. Therefore, three predicates were chosen, which should manage these states during runtime. The used predicates are shown in the following code. Both predicates state/1 and counter/1 where only used with one Variable, so that the state is first retracted and then newly asserted.

Dynamic Predicates

```
:- (dynamic collection/1).
:- (dynamic state/1).
:- (dynamic counter/1).

main:- helpsubway().

% listed with all selected items
collection(nothing).
% state for asking the reight questions
state(breads).
% state for toppings that can be choosen multiple times
counter(0).
```

Since the list is to be changed with selected(0), the central state management was implemented here.

selected(0)

```
% switch state -> next selection
selected(0) :-
    state(X),
    (   X==breads
    ->  retract(state(breads)),
        assert(state(main)),
        print("Choose the main topping now!"),
        put(10), printhelpnote()
    ;   X==main
    ->  retract(state(main)),
        assert(state(veggies)),
        print("Choose the vegetables now!"),
        put(10)
    ;
    % specific case for veggies. There can be more than
      one item selected
    X==veggies
    ->
        counter(Number),
```

```
19          maxVeggies(Max),
20          (
21              (
22              % Ask for another veggie topping
23              Number < Max -> print("Do you want to choose
                    more? [y/n]"),
24              read(Like),
25                   Like==y
26              ->  retract(counter(Number)),
                    assert(counter(Number + 1)), print("OK,
                    You can choose "); true);
27              % continue with the next case, set new state
28              retract(state(veggies)),
29              assert(state(sauce)),
30              print("Choose the sauce now!"),
31              put(10)
32          )
33      ;
34      X==sauce
35       ->  retract(state(sauce)),
36          assert(state(sides)),
37          print("Choose the sides now!"),
38          put(10)
39       ;  X==sides
40       ->  retract(state(sides)),
41          assert(state(breads)),
42          done(1),
43          abolish(collection/1),
44          assert(collection(nothing)),
45          print("Thanks for eating at Subway"),
46          put(10)
47       ).
```

3 Documentation