German University in Cairo
Department of Computer Science
Dr. Nourhan Ehab

Introduction to Artificial Intelligence, Winter Term 2025
Assignment 2: AI Sous-Chef

Due December 5th by 23:59

1. Project Description

In this project, you will implement a logic-based agent that simulates a chef responsible for constructing a burger sandwich. The agent will reason about actions and states using the *Situation Calculus* and will be implemented in *Prolog*.

The system models the process of burger assembly as a reasoning task, where the agent determines the correct stacking order of ingredients based on the knowledge base (KB). Logical predicates such as `above/2` will represent the spatial relationships between ingredients, where `above(X, Y)` indicates that ingredient `X` should be above ingredient `Y` (not necessarily directly above it). The `above/2` predicate guides the agent in achieving a valid final configuration of the burger.

We make the following simplifying assumptions:

a) The goal of the agent is to ensure that every ingredient is correctly stacked to complete the burger.

b) The first ingredient will always be the *bottom bun*, and the last one will always be the *top bun*.

c) The order of the ingredients between the bottom bun and the top bun will depend on the restrictions defined in the knowledge base (KB) by the predicate `above(X, Y)`.

d) The only ingredients that we can stack are: `bottom_bun, patty, lettuce, cheese, pickles, tomato, top_bun`.

e) Ordering constraints given will always be consistent.

f) Initially, none of the ingredients are stacked.

g) The goal is to stack one only one of each ingredient.

The agent can perform the action `stack(X)` where `X` is an ingredient. This action adds one of this ingredient to the top of the burger.

To correctly implement this agent, you need to follow the following steps.

a) On the CMS, you will find two Prolog files, `KB1.pl` and `KB2.pl` containing two different sample knowledge bases with the initial state of the world.You can use these files to test your code on two different instances of the problem. Do not add anything to either file, and create a new Prolog file `burger_TeamID.pl` in which you will write your implementation, you have to name the file with your team number. `burger_TeamID.pl` must be in the same directory in which `KB.pl` lies. Import the knowledge base you want to use at the beginning of `burger_TeamID.pl` using

```
:- include('Filename.pl').
```

b) Come up with fluents (predicate symbols whose denotations change across situations) to describe the state of the world. For each fluent, write a successor-state axiom in `burger_TeamID.pl`. Whenever possible, it is preferable to use built-in predicates rather than defining your own. You are free to write any helper predicates you need.

c) Write a predicate `burgerReady(S)` and use it to query the agent's KB for a situation with the complete burger in the right order. You are not required to return an optimal plan. The result of the query should be a situation described as the result of doing some sequence of actions from the initial situation s_0 (as shown in the examples in the next section).

Important Note: You might write your successor state axioms correctly, yet when you query your KB, your program might run forever. This is because Prolog uses DFS to implement backward chaining, and we know that DFS is incomplete. To solve this issue, consider using the built-in predicate `call_with_depth_limit` (`http://www.swi-prolog.org/pldoc/man?predicate=call_with_depth_limit/3`). This predicate does depth limited search to back-chain on the query provided as the first argument of the predicate. You can use this built-in predicate to implement another predicate to do iterative deepening search when solving for `burgerReady(S)`. In this way, you will guarantee that you will reach a solution (if there is one) since iterative deepening search is complete.

Here is one way to implement IDS in Prolog. (You can write it in any way you like.)

```
ids(X,L):-
(call_with_depth_limit(myPredicate(X),L,R), number(R));
(call_with_depth_limit(myPredicate(X),L,R), R=depth_limit_exceeded,
                                            L1 is L+1, ids(X,L1)).
```

2. Example

- **Knowledge base (KB1.pl)**:

```
above(tomato, lettuce).
above(tomato, patty).
above(pickles, cheese).
above(cheese, tomato).
```

- **Query1**: burgerReady(S).
- **Output1**: S = result(stack(top_bun), result(stack(pickles), result(stack(cheese), result(stack(tomato), result(stack(patty), result(stack(lettuce), result(stack(bottom_bun), s0)))))))
- **Query2**: burgerReady(result(stack(top_bun), result(stack(pickles), result(stack(cheese), result(stack(tomato), result(stack(patty), result(stack(lettuce), result(stack(bottom_bun), s0)))))))).
- **Output2**: true.
- **Query3:** burgerReady(result(stack(top_bun), result(stack(pickles), result(stack(cheese), result(stack(tomato), result(stack(lettuce), result(stack(patty), result(stack(bottom_bun), s0)))))))).

- **Output3:** true .
- **Query4**: burgerReady(result(stack(bottom_bun), result(stack(pickles), result(stack(cheese), result(stack(tomato), result(stack(patty), result(stack(lettuce), result(stack(top_bun), s0)))))))).
- **Output4:** false.
- **Query5**: burgerReady(result(stack(top_bun), result(cheese ,result(stack(pickles), result(stack(cheese), result(stack(tomato), result(stack(lettuce), result(stack(patty), result(stack(bottom_bun), s0))))))))).
- **Output5:** false.

3. Groups: Same teams as Project 1. If you wish to change your team, you can send your TA an email.

4. Regulations

a) Your implementation must be a successor state axiom otherwise the grade will be zero.

b) Grading will be based on the public test cases (KB1, KB2), private test cases given that your implementation correctly uses situation calculuss.

c) Your code must include detailed comments explaining each fluent implemented and its arguments.

d) The use of online sources and LLM tools is allowed as an aid (not copy and paste) but the final work submitted must be done by the team members.

e) You are not allowed to use the built-in predicates `assert/1` or `retract/1` or any of their variants.

5. Submission

Source code and Project Report. On-line submission by December 5th at 23:59.

Submission form. You should submit a .zip file containing `burger_TeamID.pl` using the following form. (Note: Do not put the KB inside your Prolog file just include it.) and a `.txt` file containing your team member names and IDs.

`https://forms.gle/T9eVpXs3eLyYxYqZ7`.

Brainstorming Session. In tutorials.