**CS 6364 Artificial Intelligence**

**Project Report**

**Professor: Dr. Mithun Balakrishna**

By: Sathishbabu Rajasekar

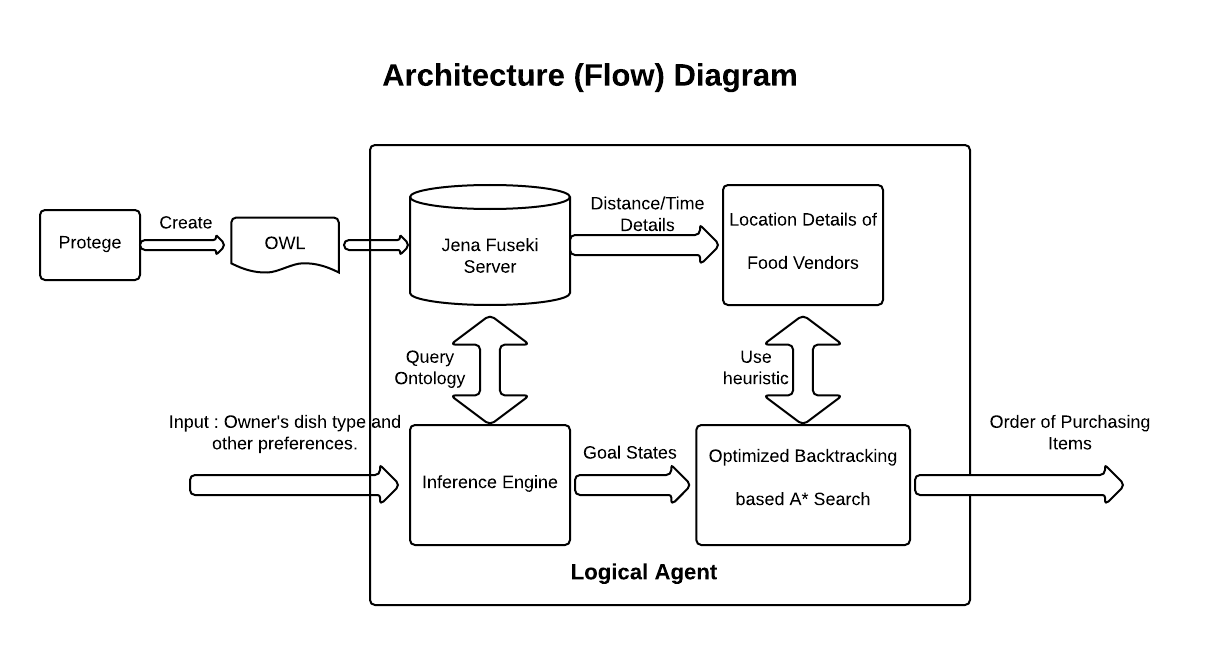
Sxr136330

**Restaurant Menu Planner and Item Purchaser**

**Problem Description:**

Implement a logical agent that will accept a Restaurant owner’s current inventory status of items and preferred type of dishes he intends to cook for the next one week .Based on the details given about the dishes, the logical agent must infer the most appropriate set of dishes that match all the details provided. After inferring the most appropriate set of dishes, the logical agent must find the ingredients used in these dishes and the vendors that sell the found ingredients. The agent should also find a travel route which gives the order in which the owner has to purchase the Items from different vendors that will satisfy the owner’s requirement. The logical agent should use the domain Knowledge and reasoning mechanism to infer the list of destination vendors that can satisfy the owner’s activities requirement and heuristics to be used in the informed search to find the optimal route to destination vendor(s) in an optimal sequence.

**Proposed Solution:**



There are three steps to the solution of the above problem,

1. Knowledge Bank: I am going to store all the KB in OWL, which the inference engine is going to use to get information regarding the dishes.
2. Inference Engine: The inference engine gets the type of the dishes from the Owner and infers the ingredients necessary to be purchased.
   1. Example:

Turkish Lamb Kofte is cooked in large quantity in the month of April. Peak summer months in America are April, May and June. Turkish Lamb Kofte is a Mediterranean dish which is made using Lamb and chilly.

Example Dish:

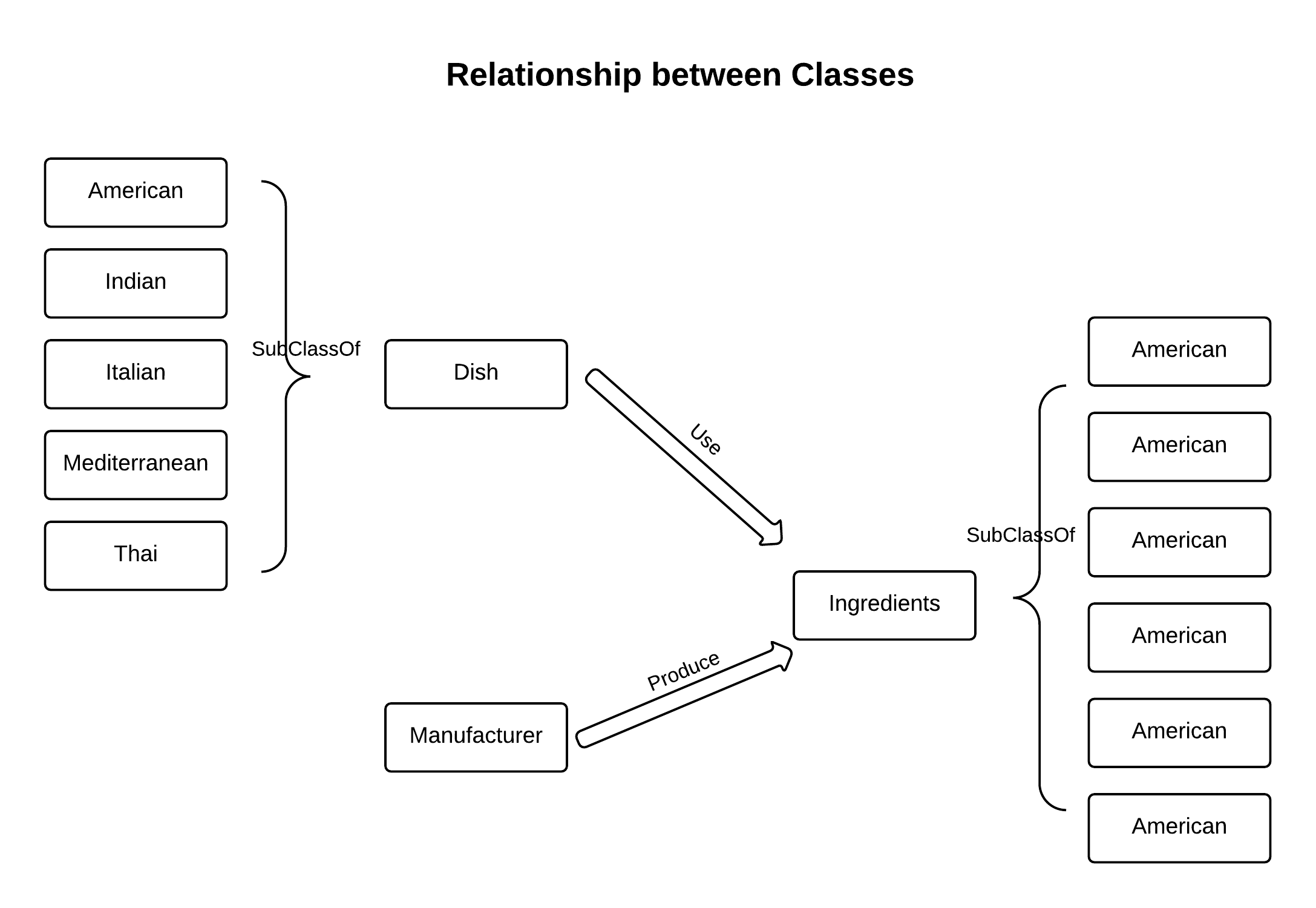
* + 1. Mediterranean Dish suitable for the hot climate.

1. Informed Search: The informed search is going to be performed using the results of the inference engine and the location details of each food vendors.

The above three parts of the problem are solved as follows,

**Knowledge Bank (KB):**

A Knowledge Bank (KB) in OWL is created using the Protégé tool. The relationship between various classes of the KB are as follows:



The Ontology has the following relationship:

1. Dish use Ingredients
2. Manufacture produce Ingredients

Also the KB is used to store the value of the actual distance between two manufactures / vendors and the value of actual time it takes to travel from on one manufacturer to other manufacturer which is used by the A\* algorithm to calculate the most optimum path for the chosen heuristic.

**Inference Engine:**

The Inference Engine used is the Apache Jena Engine. To make querying the inference engine easier, I used the Jena Fuseki Server. Using the Jena Java API, I queried the Fuseki server and obtained the necessary information’s such as:

1. Dishes which are made of Ingredients that exist in a particular form (powder, liquid etc).
2. Manufactures who sell Ingredients to make a particular dish.
3. All the Ingredients the manufacture produces.
4. The distance between two manufacturers.
5. Dishes that are spicy, sweet or sour.

And many other combination of conditions based on the user input.

The user input is obtained in the following format:

<DishType>, <Taste>, <Cooked\_for>, <Cooked\_in>, <Ingrident\_form>

, with any of the above condition being null or desired value.

Once the user input his desired conditions which the dish has to satisfy, the logical agent queries to the Inference Engine to find all possible dishes that satisfy the user’s requirements. Of all the dishes that satisfy the query, the logical agent chooses the first dish as the preferred dish. Here I am assuming that the first dish is the best dish though all dishes satisfy the user’s requirement equally to reduce the complexity of the further calculations as there are already more than one dish is involved in finding the final travel path.

**Informed Search:**

The Informed Search algorithm chosen for the above problem is the A\* Search algorithm. For the A\* algorithm, I have chosen the following heuristics for comparison:

1. Heuristic 1: Minimum distance between two different nodes.
2. Heuristic 2: Minimum time taken to travel between one nodes to another.

In my program, I have chosen total of six nodes including the starting node which is the restaurant itself. The Five nodes are the nodes that correspond to the food manufacturers, which are as follows:

1. Kellogs
2. Cargil
3. Precise food
4. Profood
5. Kraft food.

The minimum distance and minimum time between different nodes are stored as part of the class definition as these two values do not change over time .The actual distance between two nodes and the actual time taken to travel between various nodes is stored in the KB and are queried in when necessary. In addition to using the A\* algorithm, I used Backtracking to solve each constraints in the given problem.

**Examples:**

In the first example, I am trying to trying to find dish that is American or Indian dish, spicy, ate for lunch or breakfast, uses ingredients of the form liquid or cut or seeds:

Please enter the number of Dishes you want to cook: 1

Please choose from one of the following:

1. Buy Ingredients in the same order the dishes are specified.

2. Buy Ingredients in any order that gives minimum value.

2

Enter dish details of the form - <DishType>, <Taste>, <Cooked\_for>, <Cooked\_in>, <Ingrident\_form>

American Indian,spicy,lunch breakfast,summer winter,liquid cut seeds

Choosen Dish for the given details:

**Briyani**

**Using Minimum Path as Heuristic:**

Complete Path:

**resturant->Kellogs->Kraft\_food**

Detailed Path:

Go to:resturant

Go to:Kellogs and buy: Carrot

Go to:Kraft\_food and buy: Brownrice Chicken Whiterice

**Using Minimum Time as Heuristic:**

Complete Path:

**resturant->Kraft\_food->Profood**

Detailed Path:

Go to:resturant

Go to:Kraft\_food and buy: Brownrice Chicken Whiterice

Go to:Profood and buy: Carrot

It can be seen that even for a simple example with one dish, the optimum path given by the two heuristics are entirely different.

The second example is a bit more complex, with 3 dishes:

Please enter the number of Dishes you want to cook: 3

Please choose from one of the following:

1. Buy Ingredents in the same order the dishes are specified.

2. Buy Ingredients in any order that gives minimum value.

2

Enter dish details of the form - <DishType>, <Taste>, <Cooked\_for>, <Cooked\_in>, <Ingrident\_form>

American Indian,spicy,lunch breakfast,summer winter,liquid cut seeds

Choosen Dish for the given details:

**Briyani**

Enter dish details of the form - <DishType>, <Taste>, <Cooked\_for>, <Cooked\_in>, <Ingrident\_form>

Mediterranean,spicy,,,

Choosen Dish for the given details:

**Chicken\_Shwarma**

Enter dish details of the form - <DishType>, <Taste>, <Cooked\_for>, <Cooked\_in>, <Ingrident\_form>

American Indian,sour sweet,lunch breakfast,summer winter,liquid cut seeds

Choosen Dish for the given details:

**Rosemary\_Corn\_Bread**

**Using Minimum Path as Heuristic:**

Complete Path:

**resturant->Cargil->Precise\_food->Profood->Kraft\_food**

Detailed Path:

Go to:resturant

Go to:Cargil and buy: Bread

Go to:Precise\_food and buy: Chicken Oregano

Go to:Profood and buy: Brownrice Carrot Onion

Go to:Kraft\_food and buy: Whiterice

**Using Minimum Time as Heuristic:**

Complete Path:

**resturant->Kraft\_food->Precise\_food->Kellogs**

Detailed Path:

Go to:resturant

Go to:Kraft\_food and buy: Brownrice Chicken Whiterice

Go to:Precise\_food and buy: Oregano

Go to:Kellogs and buy: Carrot Onion Bread

The second example takes long time to complete as it has more number of combinations possible to generate the best possible travel path to buy the necessary ingredients.

**Programming tools:**

The following tools where used for solving the above problem,

* Jena API
* Jena Apache Fuseki Server
* Protégé
* Eclipse (Java IDE)
* OWL Ontology

**Problem Encountered:**

Since I used Backtracking I encountered the following problems:

1. Repeated calculation of the optimum path between same nodes.
2. Nodes visiting Nodes that are already in its path.

Backtracking based CSP solvers have non-polynomial running time, to reduce the running time for certain scenarios and to optimize the program to resolve the issues encountered above, I did the following changes,

1. Store and use pre-calculated optimum path information

For example, if optimum route to go from Kellogs to Cargil is already calculated, it is stored and used when calculating optimum route from Kellogs to Profood via Cargil.

1. Remove Nodes that are already part of the Optimum path

For example, if the current path is

Restaurant->Kellogs->Cargil

I removed finding the route from Kellogs to Cargil, as it is already in its optimum path, by analyzing the optimum path for existing nodes.

**Pending Issues:**

As I mentioned before, I have taken the first dish that matched all the user’s requirement and built the combination of ingredient manufactures based on it .This is valid in cases where the output of the user’s requirement is only one dish but if more than one dish matches the query , then the assumption becomes invalid. To remove this issue, we need to use one more Backtracking algorithm which finds the combination of dishes and their corresponding combination of ingredient manufactures. This significantly increases the runtime and ways to optimize it should be researched.

**Potential Improvements:**

Some of the Potential Improvements I can foresee are,

* Implemented the Solution to use more than one dish that satisfy the given user conditions.
* Currently the system is designed for a 6x6 grid of nodes; this can be made to generic any size grid.
* If we make the number of nodes be increased to any amount, we will have problems at scale, for example above 25-30 nodes. In those cases , we may have to improve the program by saving potential parts of the calculation in Files because the memory will easily run out as number of nodes increase.