Knowledge Engineering project

Msc Computer Science, Knowledge Engineering course
A.Y. 2021/2022

Aqila Farahmand - 121208
Alessio Galassi - 118576

Abstract

Nowadays, every time, people are put in front of huge amounts of data and information on which they have to make choices, and this scenario regards a wide range of situations, from the one simpler to others more relevant. With the Covid-19 pandemic have been introduced lots of new normative to decrease the spread of the virus, especially in circumstances where people are more exposed, lot of these regard all the activities performed in a restaurant. One of them is the abolition of the printed menu so that quite all the restaurants had to digitalise it, but presenting a big list of dishes on a discrete size screen it's not always easy. In this project, we propose a personalized menu that represents the knowledge about meals and guest preferences and creates a system that allows the selection of those meals that fit the guest's preferences. We try to module the system using three different methods, the Decision tables, Prolog and Knowledge graph, and then we compare the advantages and disadvantages of the three knowledge-based solutions.

1 Introduction

Technology is evolving rapidly. Artificial intelligence and other advancements in information technology specifically set the stage for more technological evolution. Hard copies of restaurants are becoming archaic and will be a thing of the past in the next few years. Replacing the hard copies of restaurant menus with the digital menu will help you in ways more than one. Not only is it cost-effective since you will not have to print menus every time you bring in the slightest chance, but it is also very appealing to the eyes since your customers can see the entire menu in one sight, which in turn will help you in upselling, engaging and understanding the customers better, However, some guests do not want every meal, e.g. vegetarians or guests with an allergy. Using a knowledge base system, instead of showing all the meals that are offered, we can create a personalized menu where the customers can sort the menu based on their own preferences to help them choose the dishes that fit them.
In this project, we propose a personalized menu that represents the knowledge about meals and guest preferences and creates a system that allows the selection of those meals that fit the guest's preferences. We try to module the system using three different methods, the Decision tables, Prolog and Knowledge graph, and then we compare the advantages and disadvantages of the three knowledge-based systems.

2 Decision tables

Decision tables are a declarative solution to manage business logic only specifying what needs to be done, with no details as to how to, in a step-by-step manner, must be carried out; differently from a procedural way. Decision tables and decision rules are declarative as they prescribe decision criteria namely conditions and not tasks, that are tested without a specific order. Decision Model and Notation, DMN, is a standard from Object Management Group, OMG, that has the purpose of providing the constructs that are needed to model decisions, so that organizational decision-making can be readily depicted in diagrams, accurately defined by business analysts and if needed automated. The project has used Camunda Modeler, a tool to design BPMN(Business Process Model and Notation) and indeed DMN diagrams, developed by Camunda company. An entry point is been considered the input is given by the guest, i.e. his preferences or intolerances, that constitute the base for the first table where the rule that matches the inputs returns a list of ingredients suitable for the specific guest. The second table collects all the rows where the ingredients that compose a dish are contained in the list of ingredients allowed for the guest, returning a list of dishes that are on the filtered menu according to the inputs specified in the first step.

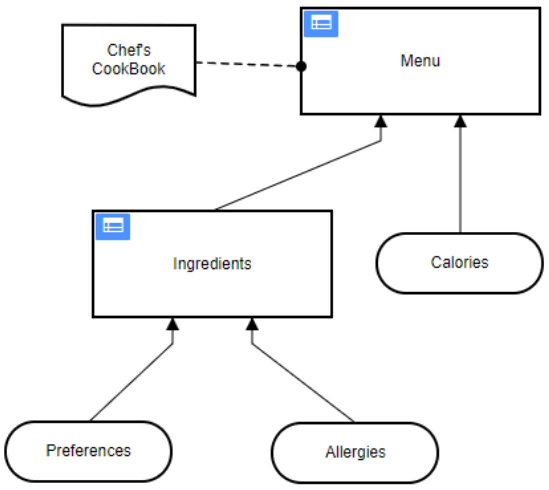
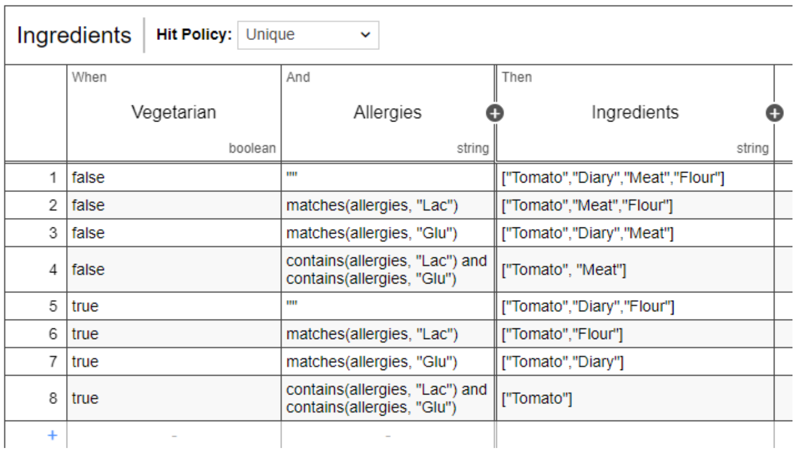


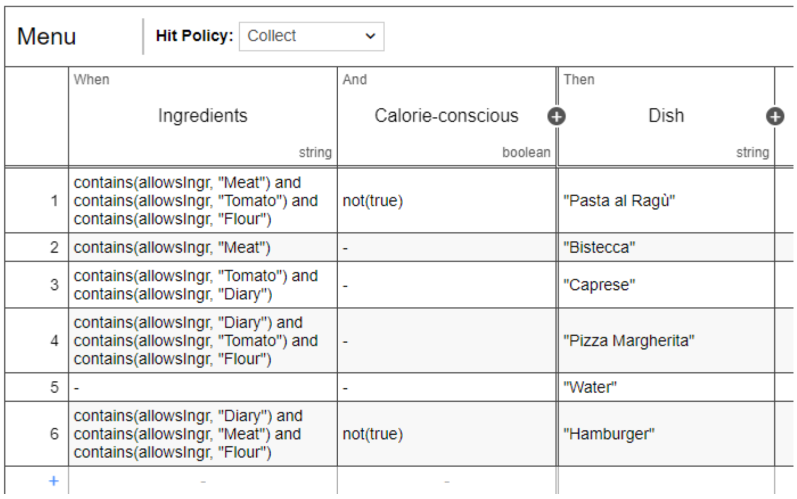
Figure 1: DMN model produced with Camunda Modeler.



Ingredients Hit Policy: Unique

	When	And	Then
	Vegetarian (boolean)	Allergies (string)	Ingredients (string)
1	false	""	["Tomato","Diary","Meat","Flour"]
2	false	matches(allergies, "Lac")	["Tomato","Meat","Flour"]
3	false	matches(allergies, "Glu")	["Tomato","Diary","Meat"]
4	false	contains(allergies, "Lac") and contains(allergies, "Glu")	["Tomato", "Meat"]
5	true	""	["Tomato","Diary","Flour"]
6	true	matches(allergies, "Lac")	["Tomato","Flour"]
7	true	matches(allergies, "Glu")	["Tomato","Diary"]
8	true	contains(allergies, "Lac") and contains(allergies, "Glu")	["Tomato"]
+	-	-	

Figure 2: First table.



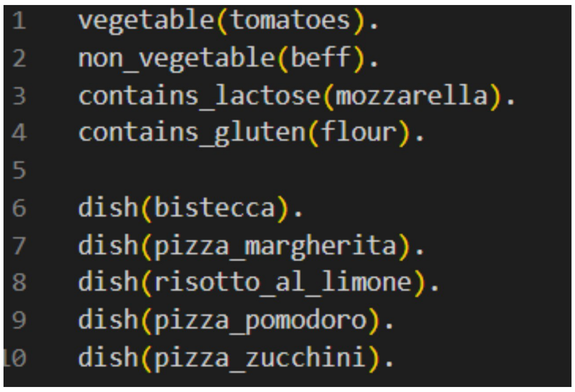
Menu | Hit Policy: Collect

	When Ingredients string	And Calorie-conscious boolean	Then Dish string
1	contains(allowsIngr, "Meat") and contains(allowsIngr, "Tomato") and contains(allowsIngr, "Flour")	not(true)	"Pasta al Ragù"
2	contains(allowsIngr, "Meat")	-	"Bistecca"
3	contains(allowsIngr, "Tomato") and contains(allowsIngr, "Diary")	-	"Caprese"
4	contains(allowsIngr, "Diary") and contains(allowsIngr, "Tomato") and contains(allowsIngr, "Flour")	-	"Pizza Margherita"
5	-	-	"Water"
6	contains(allowsIngr, "Diary") and contains(allowsIngr, "Meat") and contains(allowsIngr, "Flour")	not(true)	"Hamburger"
+	-	-	

Figure 3: Second table.

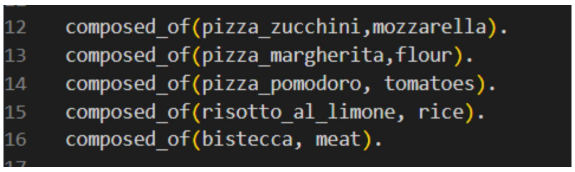
3 Prolog

Prolog is a logic programming language associated with artificial intelligence and computational linguistics. Prolog has its roots in first-order logic, a formal logic, and unlike many other programming languages, Prolog is intended primarily as a declarative programming language: the program logic is expressed in terms of relations, represented as facts and rules. A computation is initiated by running a query over these relations. In Prolog we describe a situation and based on this code, the interpreter or compiler will tell us a solution. The computer will tell us whether a Prolog sentence is true or not and if it contains variables, what the values of the variables need to be. Prolog is most useful in the areas related to artificial intelligence research, such as problem-solving, (path) planning or natural language interpretation. Prolog first appeared in 1972, whereas java in 1995, python in 1991, and C in 1972. Prolog has its roots in first-order logic (also known as predicate logic) which are objects, facts and relations. Usually, there is a collection of sentences that are assumed to be true, to create a logical definition of predicates. Such a collection of sentences that are true is called a knowledge base. SWI-Prolog is a versatile implementation of the Prolog language that we have used to implement our project. It aims at scalability. Its robust support for multi-threading exploits multi-core hardware efficiently and simplifies embedding in concurrent applications.
To make the model as simple as possible we have considered a dish with a few ingredients, as the objective is to create a prototype model before actual application in the real world. First, we define the facts or predicates as follows:



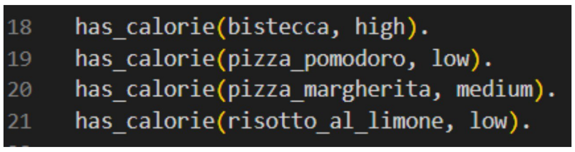
```
vegetable(tomatoes).
non_vegetable(beff).
contains_lactose(mozzarella).
contains_gluten(flour).

dish(bistecca).
dish(pizza_margherita).
dish(risotto_al_limone).
dish(pizza_pomodoro).
dish(pizza_zucchini).
```

The relation between the predicates is defined, i.e we define the relationship between the dish and the ingredients it is composed of:



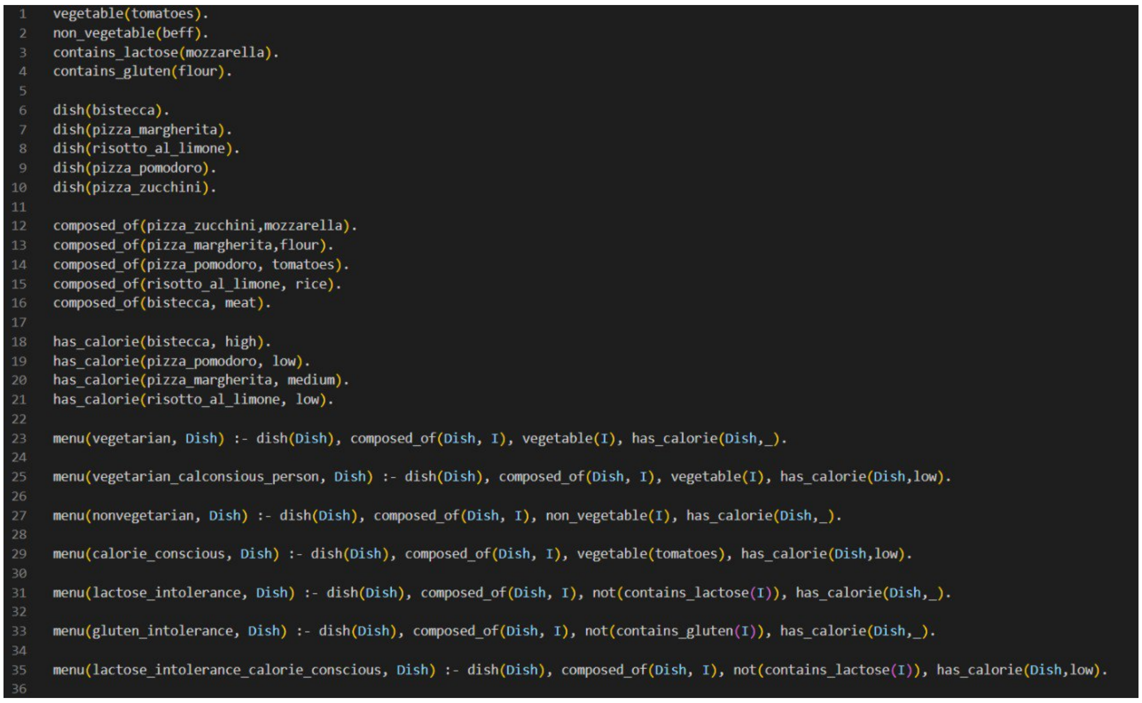
```
composed_of(pizza_zucchini,mozzarella).
composed_of(pizza_margherita,flour).
composed_of(pizza_pomodoro, tomatoes).
composed_of(risotto_al_limone, rice).
composed_of(bistecca, meat).
```

We also define the has_calorie relation to indicate the information about the calorie level for the dishes:



```
has_calorie(bistecca, high).
has_calorie(pizza_pomodoro, low).
has_calorie(pizza_margherita, medium).
has_calorie(risotto_al_limone, low).
```

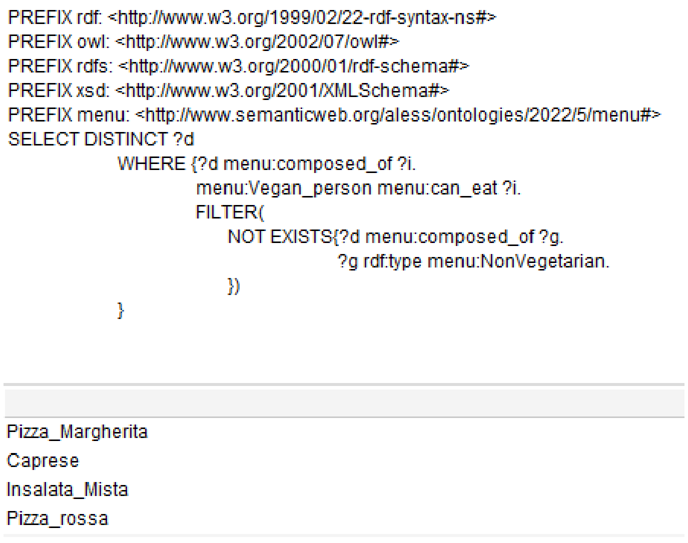
Finally, we define the rule for each type of customer, that is, each customer can set their preference and according to which a dish that fits the customer preference is suggested to the customer.



```
vegetable(tomatoes).
non_vegetable(beff).
contains_lactose(mozzarella).
contains_gluten(flour).

dish(bistecca).
dish(pizza_margherita).
dish(risotto_al_limone).
dish(pizza_pomodoro).
dish(pizza_zucchini).

composed_of(pizza_zucchini,mozzarella).
composed_of(pizza_margherita,flour).
composed_of(pizza_pomodoro, tomatoes).
composed_of(risotto_al_limone, rice).
composed_of(bistecca, meat).

has_calorie(bistecca, high).
has_calorie(pizza_pomodoro, low).
has_calorie(pizza_margherita, medium).
has_calorie(risotto_al_limone, low).

menu(vegetarian, Dish) :- dish(Dish), composed_of(Dish, I), vegetable(I), has_calorie(Dish,_).

menu(vegetarian_calconsious_person, Dish) :- dish(Dish), composed_of(Dish, I), vegetable(I), has_calorie(Dish,low).

menu(nonvegetarian, Dish) :- dish(Dish), composed_of(Dish, I), non_vegetable(I), has_calorie(Dish,_).

menu(calorie_conscious, Dish) :- dish(Dish), composed_of(Dish, I), vegetable(tomatoes), has_calorie(Dish,low).

menu(lactose_intolerance, Dish) :- dish(Dish), composed_of(Dish, I), not(contains_lactose(I)), has_calorie(Dish,_).

menu(gluten_intolerance, Dish) :- dish(Dish), composed_of(Dish, I), not(contains_gluten(I)), has_calorie(Dish,_).

menu(lactose_intolerance_calorie_conscious, Dish) :- dish(Dish), composed_of(Dish, I), not(contains_lactose(I)), has_calorie(Dish,low).

```

4 Knowledge graph

The knowledge graph is a knowledge base that uses a graph-structured data model to store data, in this project we focused on Resource Description Framework (RDF), which is originally designed as a data model for metadata, which is used as a general method for description and exchange of graph data. RDF is based on the statement, aka triples, that associates the following elements "object-attribute-value" to build the data source. The fundamental concepts of RDF are resources, properties and statements. A resource can be viewed as an object, or as an instance where every resource has a Universal Resource Identifier (URI) that can be an URL (Web address) or some other kind of unique identifier. In the graph, every node is a resource. A property describes relations between resources, but properties themselves are a special kind of resources so they are also identified by URIs. URIs give two principal advantages: a global, worldwide, unique naming scheme, and it reduces the homonym problem of distributed data representation. Statements assert the properties of resources, it is an object-attribute-value triple that consists of a resource, a property, and a value. RDF is a universal language that lets users describe resources in their vocabulary but it doesn't assume, nor does it define the semantics of any particular application domain, to do that it is flanked by RDF Schema, which uses classes and properties. The class hierarchies, inheritance and property hierarchies are to develop reasoning over the data source. The model includes three classes: Dish, Ingredient with many subclasses like Gluten, Lactose, NonVegetarian, and lastly, the class Preference with the subclasses LactoseIntolerance, GlutenIntolerance and Vegan. Also, we have defined some object properties to correlate classes between each other, "composed_of" from Dish to Ingredient, "can_eat" from Preference to Ingredient and its inverse "is_allowed_to", finally "is_eaten_by" and "not_eaten_by" from Dish to Preference. To obtain the menu is necessary to run queries over the data, and the following query shows how to obtain the meals suitable for a vegetarian guest:



```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX menu: <http://www.semanticweb.org/aless/ontologies/2022/5/menu#>
SELECT DISTINCT ?d
          WHERE {?d menu:composed_of ?i.
                 menu:Vegan_person menu:can_eat ?i.
                 FILTER(
                    NOT EXISTS{?d menu:composed_of ?g.
                              ?g rdf:type menu:NonVegetarian.
                    })
          }
```

```
Pizza_Margherita
Caprese
Insalata_Mista
Pizza_rossa
```

Figure 4: SPARQL query.

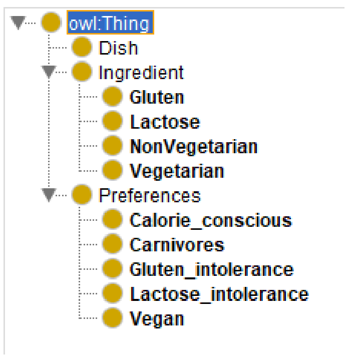


Figure 5: Classes hierarchies.