[KHAN FAMILY]

[Project Report]

GROUP MEMBERS:

1. KIRAN ZAFAR (BSCS-2019-36)
2. FAREEHA BANO (BSCS-2019-23)
3. NIDA KHAN (BSCS-2019-71)
4. MUTAHARA GHULAM (BSCS-2019-14)
5. RIJA CHOUDHARY (BSCS-2019-13)
6. ABDUL MOEEZ (BSCS-2019-08)
7. QAISAR HUSSAIN (BSCS-2019-31)

Table of Contents

Abstract:

This project explores the relationship of the specified family i.e. Khan Family. It has the ability to explore any relationship for a person. The possible relationships are baap, beti, beta, dada, nana, dadi, nani, sala, bahu, pota, nawasa, sussar, chachataya, khala, baapDada. For this task, we have used *"Python"* as frontend building while *"Prolog"* as the backend system. Python is a general-purpose and high-level programming language. It is an interpreted language. Prolog, on the other hand, is a programming language for logic. Its primary purpose is to be used as a declarative programming language. Artificial intelligence and computational linguistics are two fields where it can be used.

Predicate Logic:

Charles Pierce and Gottlob Frege are the inventors of predicate logic. Predicate logic can also be known as first-order logic. By using predicate logic, we can represent our knowledge in computer for Artificial Intelligence. We can develop information about the objects in a much simpler way and the relationship between objects can be easily expressed. We can easily capture the actual meaning of statements by using predicate logic as compared to propositional logic.

Prolog:

Prolog is a high-level computer programming language first devised for artificial intelligence applications. It is designed by Alain Colmerauer and a team of researchers. The major purpose of prolog is to use logic to represent the knowledge and to write the computer programs in the late 1970s. Prolog uses a subset of predicate logic and draws its structures from the early work of logicians. Prolog is been largely used for logic programming. Prolog is intended primarily as a declarative programming language. It expresses the logics as relations (called as facts and rules).

Prolog Features:

- ## Facts:

  Facts are those statements in prolog that describe/explains the properties of relationships between objects.

  For example: *mianbiwi (chotekhan,chotirani).*
  In the above given example "*chotekhan*" and "*chotirani*" are two objects while "*mianbiwi*" is a relation between these two objects.

- ## Terms:

  All types of data are referred to as term in prologue. For example, parent (WX, YZ) is considered a complex term made of of simple terms WX and YZ.

- ## Variables:

  They're prologue phrases that begin with capital letters or an underscore "_" . ? - mianbiwi (Y, barrirani).

  "Y" is a variable.

- ## Queries:

  A query is essentially a request for data to be retrieved from a database.

  e.g. ? **-** *parent (Y, burhan).*

  Ans: *nadir, nahid.*

- Rules:

  We can extract new properties from previously defined facts in the knowledge base using rules.

  dada(X,Y) :- parent(X,Z) , parent(Z,Y) , gins(male,X),gins(male,Z).

Problem Statement:

In this project, we have to code khan's family tree by using provided facts, and then we must implement rules given in the requirements document. The application should be able to ask few questions like, "*Who is the dada of X, who is khala of X",... etc.*

Khan-Family Tree:

Khan family tree that has been used for this project is given below:

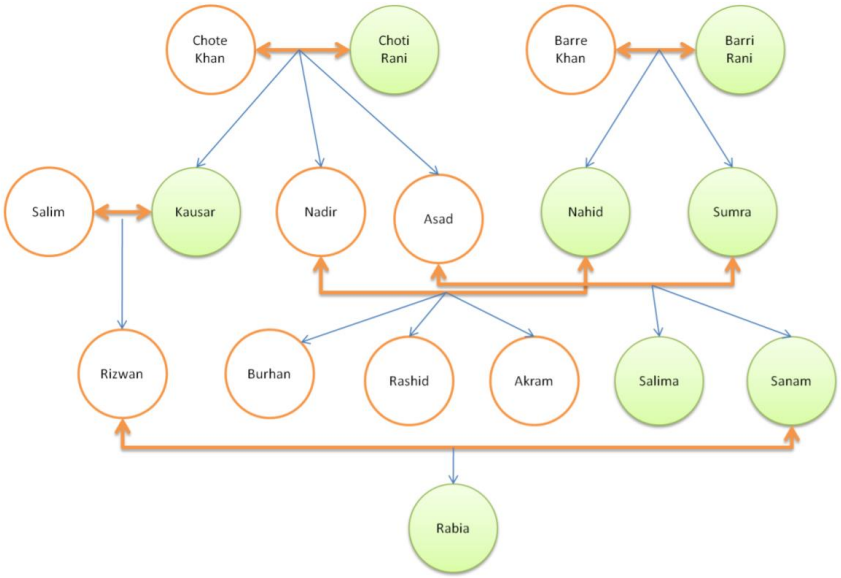

Figure 1: khan family tree

Back-end Development:

Allowed facts:

Following are the only facts that were allowed:

- mianbiwi (fact1, fact2). % fact1 is mian of fact2, who is biwi
- parent (fact1, fact2). % fact1 is parent of fact2
- gins (fact1, fact2). % fact1 can either be mard or aurat, fact2 is the name of the person

Required rules:

We have to implement the given rules in the requirements document by using the above facts.

Rules:

- *beti (X, Y): - parent (Y, X), gins (female, X).* This rule finds the daughter if we provide parent to this rule. i.e., "X" is the daughter of "Y" and we can find this by using the fact "*parent (Y, X)"* and *gins (female, X).*
- *beta (X, Y): - parent (Y, X), gins (male, X). This rule finds the son if we provide parents to this rule. i.e., "X" is the son of "Y" and we can find this by using the fact "parent (Y, X)" and gins (male, X).*
- *dada (X, Y): - parent (X, Z), parent (Z, Y), gins (male, X), gins (male, Z). This rule finds the dada (Grandpapa) if we provide pota/poti to this rule. i.e., "X" is dada of "Y" and we can find this by getting parent of the parent of Y giving the condition that all parents are male.*
- *nana (X, Y): - parent (X, Z), parent (Z, Y), gins (male, X), gins (female, Z).* This rule finds the nana if we provide nawasaa/nawasii and vice versa. In simple words find the female parent ("Z") of child "Y" and then find the male parent of mama ("Z").
- *dadi (X, Y): - parent (X, Z), parent (Z, Y), gins (female, X), gins (male, Z).* This rule finds the dadi if we provide pota/poti and vice versa. In simple words find male parent ("Z") of child "Y" and then find the female parent of papa ("Z").
- *nani (X, Y): - parent (X, Z), parent (Z, Y), gins (female, X), gins (female, Z).* This rule finds the nani if we provide nawasaa/nawasii and vice versa. In simple words find the female parent ("Z") of child "Y" and then find the female parent of mama ("Z").
- *sala (X, Y): - mianbiwi (Y, Z), parent (A, Z), gins (female, Z), parent (A, X), gins (male, A), gins (male, X).* This rule finds the sala i.e., brother of the wife. Find the
- wife of the husband ("Y") and then find the parents of the wife, after this gets all the male child of wife parent.
- *bahu (X, Y): - parent (Y, Z), gins (female, X), gins (male, Z), mianbiwi (Z, X).* Bahu is the wife of the son. To find bahu, we first son of "Y" and then find the wife of the son.
- *pota (X, Y): - parent (Y, Z), parent (Z, X), gins (male, X), gins (male, Z).* It is simple, find the son of the son.
- *nawasa (X, Y): - parent (Y, Z), parent (Z, X), gins (male, X), gins (female, Z).* Finds daughter's son.
- *sussar (X, Y): - mianbiwi (Y, Z), parent (X, Z), gins (male, X), gins (female, Z), gins (male, Y).*
- *sussar (X, Y): - mianbiwi (Z, Y), parent (X, Z), gins (male, X), gins (male, Z), gins (female, Y).* This finds the husband's father (or) wife's father.

- *baapDada (X, Y): - parent (X, Y), gins (male, X).*
- *baapDada (X, Y): - parent (X, Z), baapDada (Z, Y), gins (male, Z), gins (male, X).* This finds ancestors of a given child. This is a recursive rule that goes up the tree till we reach the top.
- *khala (X, Y): - parent (Z, Y), gins (female, Z), parent (L, Y), gins (male, L), parent (A, Z), gins (male, A), parent (A, X), gins (female, X), not (mianbiwi (L, X)).* This rule finds khala "sister(s) of mother" of a child. The main logic in this rule is that find female children of nana and then exclude son's / daughter's mother by using "*not (mianbiwi (papa, khala))"* i.e., all-female children of nana who are not the wife of child's ("Y") father.
- *chachataya (X, Y): - parent (Z, Y), gins (male, Z), parent (Ma, Y), gins (female, Ma), parent (A, Z), gins (male, A), parent (A, X), gins (male, X), not (mianbiwi (X, Ma)).* This rule finds the chaha/Taya "brother (s) of the father" of a child. The main logic in this rule is that find male children of dada and then exclude son's / daughter's father by using "*not (mianbiwi (chacha, mama))"* i.e., all male children of dada who are not husbands of child's ("Y") mother.

Front End Documentation:

In this section, we'll look at how to use prologue logic effectively using the interface. For querying using the interface, we used prologue and python.

PySwip:

PySwip is a Python library that acts as a SWI-Prolog bridge, allowing Python programmes to query Prolog. It offers us with a utility that allows us to query Prolog's backend via a Python interface.

Main functions used in Integrated environment:

➢ **Prolog.consult("filename.pl"):**

This function connects the python programme to the backend prologue file. The backend prologue file contains facts and rules that can be queried for specific results based on rules.

➢ **Prolog.query("query"):**

Prolog.query("query") sends a query to the backend prologue file and returns the query's response. In our scenario, we pass "Y" as an input to that function and save the result in the "val" array.

Results:

We created a prologue file for the backend and used PySwip to integrate it with the Python front end. We've created a straightforward terminal interface. A user can ask on relations specified in the program by following the instructions on the interface. Users can look up the "Khan Family's" relatives. Figure 2 depicts a ready-to-use interface for receiving input.

```
<<<<<<<<<<<<<<<<  WELL COME to Chat Bot  >>>>>>>>>>>>>>>>
<       Enter 1 for Dada                                 >
<       Enter 2 for Dadi                                 >
<       Enter 3 for Bahu                                 >
<       Enter 4 for Sala                                 >
<       Enter 5 for Nana                                 >
<       Enter 6 for Nani                                 >
<       Enter 7 for Khala                                >
<       Enter 8 for BaapDada                             >
<       Enter 9 for Nawasa                               >
<       Enter 10 for Pota                                >
<       Enter 11 for Beta                                >
<       Enter 12 for Beti                                >
<       Enter 13 for Pota                                >
<       Enter 0 for Exit                                 >
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

<< Enter your Choice for the Relationship that you are interested in >>
```

Figure 2: Query Interface

When a specific relation is chosen, a complete list of all members of the Khan family is displayed to assist the user in entering a valid name in order to locate the relationship as shown below:

```
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
<                                                       >
<  Following are the Khan Family Memebers:              >
<                                                       >
<  chotekhan, chotirani, barrekhan, barrirani           >
<  salim, kausar, nadir, asad, nahid, sumra             >
<  rizwan, burhan, rashid, akram, salima, sanam, rabia  >
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>


Enter name of person whose Dada is required : akram

>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Mr.chotekhan is the dada of akram.
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

<<<<<<<<<<< Enter C to Continue and S to Stop >>>>>>>>>>>
```

Figure 3: Query for Dada

In Figure 3, the user wished to find the dada of "akram" so he entered the pota name as an input, and the name of the dada appeared on the interface in response. It also ask for continue or stop to find the relation. Figure 4 shows another query and its corresponding response.

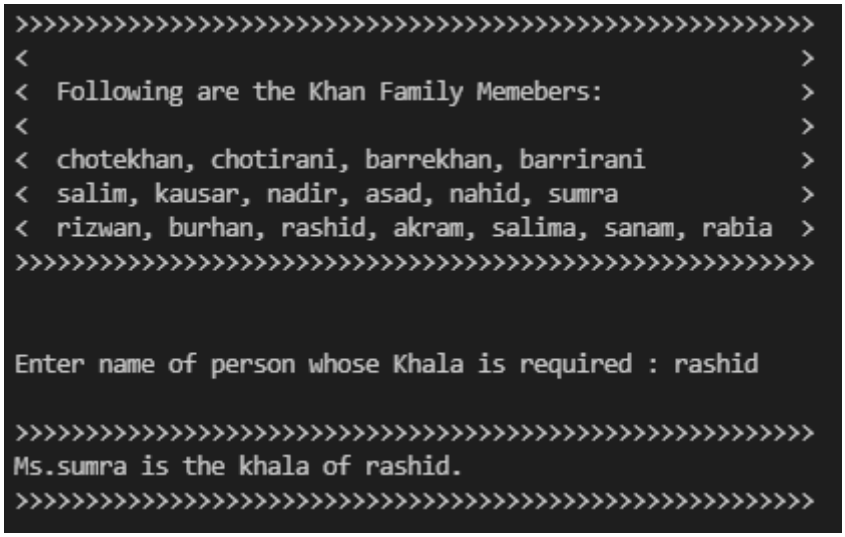

Figure 4:Query for Khala