



De La Salle University - Manila  
College of Computer Studies

**MCO2: ChatBot**

In Partial Fulfillment of the Course Requirements in CSINTSY - S17  
Term 1, AY 2023 - 2024

**Submitted by:**

**GROUP : Intzy**

Gutierrez, Allen Andrei D.

Palacios, Jeremy Koen K.

Sabater, Cassandra Jo T.

Thay, Jerrel Chadler O.

**Submitted to:**

MR. TIAM-LEE, THOMAS JAMES

November 28, 2023

## I. INTRODUCTION

Intelligent systems have shown their potential and use in today's world as it is convenient, entertaining and practical. An example of these systems would be the famous chatbot, ChatGPT. It provides users with valuable insight and information with the data users and the developers feed it.

In this premise, the goal of the researchers is to make a chatbot that focuses on answering questions and accepting statements on familial relations, through the use of Python and a knowledge base. The report will discuss the chatbot implementation and its results, the knowledge base, limitations and challenges.

## II. KNOWLEDGE BASE

This section lists and describes the rules defined in the Prolog knowledge base, representing them in first order logic.

1.  $\text{father}(X, Y) \text{ :- male}(X), \text{parent}(X, Y), X \neq Y.$ 
  - $\forall X \forall Y (\text{male}(X) \wedge \text{parent}(X, Y) \rightarrow \text{father}(X, Y))$
  - The rule encapsulates the characteristics of a father, if X or the person in question is a male and is a parent of child Y, then X is a father with a precaution that the father can't be the father of himself.
2.  $\text{mother}(X, Y) \text{ :- female}(X), \text{parent}(X, Y), X \neq Y.$ 
  - $\forall X \forall Y (\text{female}(X) \wedge \text{parent}(X, Y) \rightarrow \text{mother}(X, Y))$
  - The rule encapsulates the characteristics of a mother, if X or the person in question is a female and is a parent of child Y, then X is a mother with a precaution that the mother can't be the mother of herself.
3.  $\text{child}(X, Y) \text{ :- parent}(Y, X).$ 
  - $\forall X \forall Y (\text{parent}(Y, X) \rightarrow \text{child}(X, Y))$
  - The rule encapsulates the characteristics of a child, as long as X or the person in question has a parent or if parent Y is the parent of child X, then X is a child of parent Y.
4.  $\text{son}(X, Y) \text{ :- male}(X), \text{child}(X, Y).$ 
  - $\forall X \forall Y (\text{male}(X) \wedge \text{child}(X, Y) \rightarrow \text{son}(X, Y))$
  - Similar to the child, as long as X or the person in question has a parent or if parent Y is the parent of child X (checked through  $\text{child}(X, Y)$ ) and X is a male, then X is a son of parent Y.
5.  $\text{daughter}(X, Y) \text{ :- female}(X), \text{child}(X, Y).$ 
  - $\forall X \forall Y (\text{female}(X) \wedge \text{child}(X, Y) \rightarrow \text{daughter}(X, Y))$

- Similar to the child, as long as X or the person in question has a parent or if parent Y is the parent of child X (checked through  $\text{child}(X, Y)$ ) and X is a female, then X is a daughter of parent Y.
6.  $\text{sibling}(X, Y) :- \text{parent}(P, X), \text{parent}(P, Y), X \neq Y.$
- $\forall X \forall Y ((\exists P (\text{parent}(P, X) \wedge \text{parent}(P, Y) \wedge X \neq Y)) \rightarrow \text{sibling}(X, Y))$
  - The rule encapsulates the characteristics of a sibling, if person X and Y have the same parent P and X and Y are not the same person, then X is the sibling of Y.
7.  $\text{sister}(X, Y) :- \text{female}(X), \text{sibling}(X, Y).$
- $\forall X \forall Y (\text{female}(X) \wedge \text{sibling}(X, Y) \rightarrow \text{sister}(X, Y))$
  - Similar to siblings, if person X and Y have the same parent P and X and Y are not the same person (checked through  $\text{sibling}(X, Y)$ ) and X is a female, then X is a sister of Y.
8.  $\text{brother}(X, Y) :- \text{male}(X), \text{sibling}(X, Y).$
- $\forall X \forall Y (\text{male}(X) \wedge \text{sibling}(X, Y) \rightarrow \text{brother}(X, Y))$
  - Similar to siblings, if person X and Y have the same parent P and X and Y are not the same person (checked through  $\text{sibling}(X, Y)$ ) and X is a male, then X is a brother of Y.
9.  $\text{genderless\_uncle\_aunt}(X, Y) :- \text{parent}(P, Y), \text{sibling}(X, P).$
- $\forall X \forall Y (\exists P (\text{parent}(P, Y) \wedge \text{sibling}(X, P)) \rightarrow \text{genderless\_uncle\_aunt}(X, Y))$
  - The rule encapsulates the characteristics of a genderless aunt or uncle, if person Y has a parent P and P is siblings with person X, then X is an aunt or uncle of Y.
10.  $\text{uncle}(X, Y) :- \text{male}(X), \text{genderless\_uncle\_aunt}(X, Y).$
- $\forall X \forall Y (\text{male}(X) \wedge \text{genderless\_uncle\_aunt}(X, Y) \rightarrow \text{uncle}(X, Y))$
  - Similar to the genderless uncle or gender rule, if person Y has a parent P and P is siblings with person X (checked through  $\text{genderless\_uncle\_aunt}(X, Y)$ ) and X is a male, then X is an uncle of Y.
11.  $\text{aunt}(X, Y) :- \text{female}(X), \text{genderless\_uncle\_aunt}(X, Y).$
- $\forall X \forall Y (\text{female}(X) \wedge \text{genderless\_uncle\_aunt}(X, Y) \rightarrow \text{aunt}(X, Y))$
  - Similar to the genderless uncle or gender rule, if person Y has a parent P and P is siblings with person X (checked through  $\text{genderless\_uncle\_aunt}(X, Y)$ ) and X is a female, then X is an aunt of Y.
12.  $\text{cousin}(X, Y) :- \text{parent}(P1, X), \text{parent}(P2, Y), \text{sibling}(P1, P2), \text{not}(\text{sibling}(X, Y)), P1 \neq P2, X \neq Y.$

- $$\forall X \forall Y (\exists P1 \exists P2 (\text{parent}(P1, X) \wedge \text{parent}(P2, Y) \wedge \text{sibling}(P1, P2), \neg \text{sibling}(X, Y) \wedge P1 \neq P2 \wedge X \neq Y) \rightarrow \text{has\_two\_parents}(X, Y))$$
  - The rule encapsulates the characteristics of a cousin, if person X has a parent P1, person Y has a parent P2, P1 and P2 are siblings, X and Y are not siblings and X and Y are not the same person. then X and Y are cousins.
- 13.  $\text{cousin}(X, Y) :- \text{grandparent}(G, X), \text{grandparent}(G, Y), \neg \text{sibling}(X, Y), X \neq Y.$ 

$$\forall X \forall Y (\exists G (\text{grandparent}(G, X) \wedge \text{grandparent}(G, Y) \wedge \neg \text{sibling}(X, Y) \wedge X \neq Y) \rightarrow \text{cousin}(X, Y))$$
  - This rule for cousins is similar to the previous rule on cousins but this rule checks if the parents are cousins.
- 14.  $\text{grandparent}(X, Y) :- \text{parent}(X, Z), \text{parent}(Z, Y).$ 
  - $\forall X \forall Y (\exists Z (\text{parent}(X, Z) \wedge \text{parent}(Z, Y)) \rightarrow \text{grandparent}(X, Y))$
  - The rule encapsulates the characteristics of a grandparent as the rule checks if there exists a person Z that has a child Y and a parent X once these are satisfied then it can be concluded that X is the grandparent of Y.
- 15.  $\text{grandmother}(X, Y) :- \text{female}(X), \text{grandparent}(X, Y).$ 
  - $\forall X \forall Y (\text{female}(X) \wedge \text{grandparent}(X, Y) \rightarrow \text{grandmother}(X, Y))$
  - Similar to grandparent, the rule checks if there exists a person Z that has a child Y and a parent X and X is a female (checked through grandparent(X, Y)) once these are satisfied then it can be concluded that X is the grandmother of Y.
- 16.  $\text{grandfather}(X, Y) :- \text{male}(X), \text{grandparent}(X, Y).$ 
  - $\forall X \forall Y (\text{male}(X) \wedge \text{grandparent}(X, Y) \rightarrow \text{grandfather}(X, Y))$
  - Similar to grandparent, the rule checks if there exists a person Z that has a child Y and a parent X and X is a male (checked through grandparent(X, Y)) once these are satisfied then it can be concluded that X is the grandfather of Y.
- 17.  $\text{spouse}(X, Y) :- \text{parent}(X, C), \text{parent}(Y, C), X \neq Y.$ 
  - $\forall X \forall Y (\exists C (\text{parent}(X, C) \wedge \text{parent}(Y, C)) \rightarrow \text{spouse}(X, Y))$
  - The rule encapsulates the characteristics of a spouse, if person X is a parent of person C and person Y is also a parent of C given that X and Y are not the same person, it can be concluded that X is indeed a spouse, a married person.
- 18.  $\text{parent\_in\_law}(X, Y) :- \text{spouse}(Y, Z), \text{parent}(X, Z).$ 
  - $\forall X \forall Y (\exists Z (\text{spouse}(Y, Z) \wedge \text{parent}(X, Z)) \rightarrow \text{parent\_in\_law}(X, Y))$
  - The rule encapsulates the characteristics of a parent-in-law, if person Y is married to person Z and X is the parent of person Z, then that would mean that X is the parent-in-law of Y.

19.  $\text{ancestor}(X, Y) :- \text{parent}(X, Y).$
- $\forall X \forall Y (\text{parent}(X, Y) \rightarrow \text{ancestor}(X, Y))$
  - Essentially the rule checks for ancestry by checking if person Y has a parent.
20.  $\text{ancestor}(X, Y) :- \text{parent}(X, Z), \text{ancestor}(Z, Y).$
- $\forall X \forall Y (\exists Z (\text{parent}(X, Z) \wedge \text{ancestor}(Z, Y)) \rightarrow \text{ancestor}(X, Y))$
  - Essentially the rule checks for ancestry by checking if person Y has a grandparent, a great-grandparent and so on.
21.  $\text{ancestor}(X, Y) :- \text{genderless\_uncle\_aunt}(X, Z), \text{ancestor}(Z, Y).$
- $\forall X \forall Y (\exists Z (\text{genderless\_uncle\_aunt}(X, Z) \wedge \text{ancestor}(Z, Y)) \rightarrow \text{ancestor}(X, Y))$
  - Essentially the rule checks for ancestry by checking if person Y has an uncle or aunt.
22.  $\text{ancestor}(X, Y) :- \text{parent\_in\_law}(X, Y).$
- $\forall X \forall Y (\text{parent\_in\_law}(X, Y) \rightarrow \text{ancestor}(X, Y))$
  - Essentially the rule checks for ancestry by checking if person Y has a parent-in-law.
23.  $\text{ancestor}(X, Y) :- \text{parent\_in\_law}(X, Z), \text{ancestor}(Z, Y).$
- $\forall X \forall Y (\exists Z (\text{parent\_in\_law}(X, Z) \wedge \text{ancestor}(Z, Y)) \rightarrow \text{ancestor}(X, Y))$
  - The rule checks for ancestry by checking if person Y has a grandparent-in-law.
24.  $\text{ancestor}(X, Y) :- \text{sibling}(Y, Z), \text{parent}(X, Z).$
- $\forall X \forall Y (\text{sibling}(Y, Z) \wedge \text{parent}(X, Z) \rightarrow \text{ancestor}(X, Y))$
  - The rule checks for ancestors of half-siblings whose parents have not yet been defined. It recognizes the parent of the sibling as an ancestor of the newly added half-sibling.
25.  $\text{has\_two\_parents}(X) :- \text{parent}(P1, X), \text{parent}(P2, X), P1 \neq P2.$
- $\forall X (\exists P1 \exists P2 (\text{parent}(P1, X) \wedge \text{parent}(P2, X) \wedge P1 \neq P2) \rightarrow \text{has\_two\_parents}(X, Y))$
  - The rule checks if a person X has 2 parents given that parent P1 and parent P2 are not the same person.
26.  $\text{has\_four\_grandparents}(X) :- \text{findall}(G, \text{grandparent}(G, X), Gs), \text{list\_to\_set}(Gs, \text{UniqueGs}), \text{length}(\text{UniqueGs}, \text{Count}), \text{Count} = 4.$
- $\forall X (\exists G1 \exists G2 \exists G3 \exists G4 (\text{grandparent}(G1, X) \wedge \text{grandparent}(G2, X) \wedge \text{grandparent}(G3, X) \wedge \text{grandparent}(G4, X) \wedge G1 \neq G2 \wedge G1 \neq G3 \wedge G1 \neq G4 \wedge G2 \neq G3 \wedge G2 \neq G4 \wedge G3 \neq G4) \rightarrow \text{has\_four\_grandparents}(X))$
  - The rule checks if a person X has all four grandparents coming from both sides of the family. If all grandparents are found it returns true otherwise it returns false.

For the following rules 27 to 37, as long as X and Y are connected in the family tree in any sort of way such as parent, child, uncle, cousin, ancestor, etc. , they are considered relatives.

27. **relative(X, Y) :- parent(X, Y); parent(Y, X), X \= Y.**
  - $\forall X \forall Y (\text{parent}(X, Y) \vee \text{parent}(Y, X) \wedge X \neq Y \rightarrow \text{relative}(X, Y))$
28. **relative(X, Y) :- parent\_in\_law(X, Y); parent\_in\_law(Y, X), X \= Y.**
  - $\forall X \forall Y (\text{parent\_in\_law}(X, Y) \vee \text{parent\_in\_law}(Y, X) \wedge X \neq Y \rightarrow \text{relative}(X, Y))$
29. **relative(X, Y) :- child(X, Y); child(Y, X), X \= Y.**
  - $\forall X \forall Y (\text{child}(X, Y) \vee \text{child}(Y, X) \wedge X \neq Y \rightarrow \text{relative}(X, Y))$
30. **relative(X, Y) :- grandparent(X, Y); grandparent(Y, X), X \= Y.**
  - $\forall X \forall Y (\text{grandparent}(X, Y) \vee \text{grandparent}(Y, X) \wedge X \neq Y \rightarrow \text{relative}(X, Y))$
31. **relative(X, Y) :- sibling(X, Y); sibling(Y, X), X \= Y.**
  - $\forall X \forall Y (\text{sibling}(X, Y) \vee \text{sibling}(Y, X) \wedge X \neq Y \rightarrow \text{relative}(X, Y))$
32. **relative(X, Y) :- genderless\_uncle\_aunt(X, Y); genderless\_uncle\_aunt(Y, X), X \= Y.**
  - $\forall X \forall Y (\text{genderless\_uncle\_aunt}(X, Y) \vee \text{genderless\_uncle\_aunt}(Y, X) \wedge X \neq Y \rightarrow \text{relative}(X, Y))$
33. **relative(X, Y) :- cousin(X, Y); cousin(Y, X), X \= Y.**
  - $\forall X \forall Y (\text{cousin}(X, Y) \vee \text{cousin}(Y, X) \wedge X \neq Y \rightarrow \text{relative}(X, Y))$
34. **relative(X, Y) :- spouse(X, Y); spouse(Y, X), X \= Y.**
  - $\forall X \forall Y (\text{spouse}(X, Y) \vee \text{spouse}(Y, X) \wedge X \neq Y \rightarrow \text{relative}(X, Y))$
35. **relative(X, Y) :- ancestor(X, Y), ancestor(Y, X), X \= Y.**
  - $\forall X \forall Y (\text{ancestor}(X, Y) \vee \text{ancestor}(Y, X) \wedge X \neq Y \rightarrow \text{relative}(X, Y))$
36. **relative(X, Y) :- ancestor(X, Z), ancestor(Y, Z), X \= Y.**
  - $\forall X \forall Y (\text{ancestor}(X, Z) \vee \text{ancestor}(Y, Z) \wedge X \neq Y \rightarrow \text{relative}(X, Y))$
37. **relative(X, Y) :- ancestor(Z, X), ancestor(Z, Y), X \= Y.**
  - $\forall X \forall Y (\text{ancestor}(Z, X) \vee \text{ancestor}(Z, Y) \wedge X \neq Y \rightarrow \text{relative}(X, Y))$
38. **check\_parent(X, Y) :- not(ancestor(Y,X)), not(parent(Y,X)), not(sibling(X,Y)), not(relative(X,Y)), not(has\_two\_parents(Y)), X \= Y.**
  - $\forall X \forall Y (\neg \text{ancestor}(Y, X) \wedge \neg \text{parent}(Y, X) \wedge \neg \text{sibling}(X, Y) \wedge \neg \text{relative}(X, Y) \wedge \neg \text{has\_two\_parents}(Y) \wedge X \neq Y \rightarrow \text{check\_parent}(X, Y))$
  - The rule checks if it is possible for person X to be asserted as a parent of Y by checking if person Y, the supposed child of the person X is not an ancestor, not a parent of X, not a sibling of X, not a relative of X and if Y does not have two parents given that X and Y is not the same person.
39. **check\_parent(X, Y) :- parent(X, Z), sibling (Z, Y), not(ancestor(Y,X)), not(has\_two\_parents(Y)), X \= Y.**
  - $\forall X \forall Y (\exists Z (\text{parent}(X, Z) \wedge \text{sibling}(Z, Y) \wedge \neg \text{ancestor}(Y, X) \wedge \neg \text{has\_two\_parents}(Y)) \rightarrow \text{check\_parent}(X, Y))$
  - The rule checks if it is possible for person X to be asserted as a parent of Y by checking if X is a parent of person Z and Z is a sibling of person Y. It also

checks if Y is not an ancestor of X and Y does not have 2 parents given that X and Y are not the same person.

40. **check\_father(X, Y) :- parent(X, Y).**

- $\forall X \forall Y (\text{parent}(X, Y) \rightarrow \text{check\_father}(X, Y))$
- If the fact that X is a parent of Y already exists in the database, X can be asserted as a father of Y.

41. **check\_father(X, Y) :- check\_parent(X, Y), not(female(X)).**

- $\forall X \forall Y (\text{check\_parent}(X, Y) \wedge \neg \text{female}(X) \rightarrow \text{check\_father}(X, Y))$
- The rule checks if person X is a father, if X is not a female and if X is a parent (checked through `check_parent(X, Y)`), then X is a father and can be asserted.

42. **check\_mother(X, Y) :- parent(X, Y).**

- $\forall X \forall Y (\text{parent}(X, Y) \rightarrow \text{check\_mother}(X, Y))$
- If the fact that X is a parent of Y already exists in the database, X can be asserted as a mother of Y.

43. **check\_mother(X, Y) :- check\_parent(X, Y), not(male(X)).**

- $\forall X \forall Y (\text{check\_parent}(X, Y) \wedge \neg \text{male}(X) \rightarrow \text{check\_mother}(X, Y))$
- The rule checks if person X is a mother, if X is not a male and if X is a parent (checked through `check_parent(X, Y)`), then X is a mother and can be asserted.

44. **check\_child(X, Y) :- check\_parent(Y, X), X \= Y.**

- $\forall X \forall Y (\text{check\_parent}(Y, X) \wedge X \neq Y \rightarrow \text{check\_child}(X, Y))$
- The rule checks if it is possible to assert person X as a child of Y by checking if person Y, the supposed parent, can be asserted as person X's parent given that X and Y are not the same person.

45. **check\_child(X, Y) :- sibling(Z, X), parent(Y, Z), not(ancestor(X, Y)), not(has\_two\_parents(X)), X \= Y.**

- $\forall X \forall Y (\text{sibling}(Z, X) \wedge \text{parent}(Y, Z) \wedge \neg \text{ancestor}(X, Y) \wedge \neg \text{has\_two\_parents}(X) \rightarrow \text{check\_child}(X, Y))$
- The rule checks if it is possible to assert person X as a child of Y by checking if Person Z is a sibling of X and that Z shares the same parent Y. It also checks if X is not an ancestor of Y and X does not have two parents given that X and Y are not the same person. Essentially it checks if the child has a sibling who has that parent to overpass relative.

46. **check\_son(X, Y) :- check\_child(X, Y), not(female(X)).**

- $\forall X \forall Y (\text{check\_child}(X, Y) \wedge \neg \text{female}(X) \rightarrow \text{check\_son}(X, Y))$
- The rule checks if a person X is a son, if X is not a female, is a child and has a parent (checked through `check_child(X, Y)`), then X is a son and can be asserted.

47. **check\_daughter(X, Y) :- check\_child(X, Y), not(male(X)).**

- $\forall X \forall Y (\text{check\_child}(X, Y) \wedge \neg \text{male}(X) \rightarrow \text{check\_daughter}(X, Y))$
- The rule checks if a person X is a daughter, if X is not a male, is a child and has a parent (checked through  $\text{check\_child}(X, Y)$ ), then X is a daughter and can be asserted.

**48.  $\text{check\_sibling}(X, Y) :- \text{not}(\text{relative}(X, Y)), X \neq Y.$**

- $\forall X \forall Y (\neg \text{relative}(X, Y) \wedge X \neq Y \rightarrow \text{check\_sibling}(X, Y))$
- The rule checks if it's possible to assert person X as a sibling of person Y by checking if X and Y are not relatives given that X and Y are not the same person.

**49.  $\text{check\_brother}(X, Y) :- \text{check\_sibling}(X, Y), \text{not}(\text{female}(X)).$**

- $\forall X \forall Y (\text{check\_sibling}(X, Y) \wedge \neg \text{female}(X) \rightarrow \text{check\_brother}(X, Y))$
- The rule checks if a person X is a brother, if X is not a female, is a child, has a parent (checked through  $\text{check\_child}(X, Y)$ ) and has siblings (checked through  $\text{check\_sibling}(X, Y)$ ), then X is a son and can be asserted.

**50.  $\text{check\_sister}(X, Y) :- \text{check\_sibling}(X, Y), \text{not}(\text{male}(X)).$**

- $\forall X \forall Y (\text{check\_sibling}(X, Y) \wedge \neg \text{male}(X) \rightarrow \text{check\_sister}(X, Y))$
- The rule checks if a person X is a sister, if X is not a male, is a child, has a parent (checked through  $\text{check\_child}(X, Y)$ ) and has siblings (checked through  $\text{check\_sibling}(X, Y)$ ), then X is a sister and can be asserted.

**51.  $\text{check\_genderless\_uncle\_aunt}(X, Y) :- \text{not}(\text{relative}(X, Y)), \text{not}(\text{ancestor}(Y, X)), X \neq Y.$**

- $\forall X \forall Y (\neg \text{relative}(X, Y) \wedge \neg \text{ancestor}(Y, X) \wedge X \neq Y \rightarrow \text{check\_genderless\_uncle\_aunt}(X, Y))$
- The rule checks if it is possible for person X to be asserted as an aunt or uncle of Y by checking if person X and Y are not relatives and Y is not an ancestor of X given that X and Y are not the same person.

**52.  $\text{check\_uncle}(X, Y) :- \text{check\_genderless\_uncle\_aunt}(X, Y), \text{not}(\text{female}(X)).$**

- $\forall X \forall Y (\text{check\_genderless\_uncle\_aunt}(X, Y) \wedge \neg \text{female}(X) \rightarrow \text{check\_uncle}(X, Y))$
- The rule checks if a person X is an uncle, if X is not a female and is a genderless uncle or aunt (checked through  $\text{check\_genderless\_uncle\_aunt}(X, Y)$ ), then X is an uncle and can be asserted.

**53.  $\text{check\_aunt}(X, Y) :- \text{check\_genderless\_uncle\_aunt}(X, Y), \text{not}(\text{male}(X)).$**

- $\forall X \forall Y (\text{check\_genderless\_uncle\_aunt}(X, Y) \wedge \neg \text{male}(X) \rightarrow \text{check\_aunt}(X, Y))$
- The rule checks if a person X is an aunt, if X is not a male and is a genderless uncle or aunt (checked through  $\text{check\_genderless\_uncle\_aunt}(X, Y)$ ), then X is an aunt and can be asserted.



54. **check\_grandparent(X, Y) :-** not(ancestor(X, Y)), not(parent(X, Y)), not(relative(X, Y)), not(has\_four\_grandparents(Y)), X \= Y.  
 $\forall X \forall Y (\neg \text{ancestor}(X, Y) \wedge \neg \text{parent}(X, Y) \wedge \neg \text{relative}(X, Y) \wedge \neg \text{has\_four\_grandparents}(Y) \wedge X \neq Y \rightarrow \text{check\_child}(X, Y))$
- The rule checks if it is possible for person X to be asserted as a grandparent of Y by checking if person X is not an ancestor of Y, not a parent of Y, not a relative of Y and if Y does not have four grandparents given that X and Y is not the same person.
55. **check\_grandfather(X, Y) :-** check\_grandparent(X, Y), not(female(X)).
- $\forall X \forall Y (\text{check\_grandparent}(X, Y) \wedge \neg \text{female}(X) \rightarrow \text{check\_grandfather}(X, Y))$
  - The rule checks if a person X is a grandfather, if X is not a female and is a grandparent (checked through check\_grandparent(X, Y)), then X is a grandfather and can be asserted.
56. **check\_grandmother(X, Y) :-** check\_grandparent(X, Y), not(male(X))
- $\forall X \forall Y (\text{check\_grandparent}(X, Y) \wedge \neg \text{male}(X) \rightarrow \text{check\_grandmother}(X, Y))$
  - The rule checks if a person X is a grandmother, if X is not a male and is a genderless uncle or aunt (checked through check\_genderlessuncle(X, Y)), then X is a grandmother and can be asserted.

### III. CHATBOT IMPLEMENTATION

#### 3.1 PRELIMINARIES

The ChatBot is implemented in Python and uses the ‘pyswip’ package to connect with the Prolog-centered knowledge base. The way the program gets data or provides answers is through asking for input from the user. This input may be a statement that can feed the chatbot more data or a question to be answered. Upon entering input, the program first determines the relation that the statement or question is referring to through the get\_relation() method, which splits the sentence into words and finds an exact match of a relation word. For instance, ‘child’ or ‘children’ will return the string ‘child’. If no relation is found, the None keyword is returned. Afterwards, it checks if the word ‘bye’ is present in the sentence to indicate the user’s wish to exit the program.

If the user does not wish to exit, the next conditional checks if a relation word exists in the program. If a relation word exists, the bot then checks if the sentence is a question or a statement, after which pattern matching takes place. In order to comply with the specified sentence formats for statements and questions, the ‘re’ module is imported in order to facilitate working with regular expressions in Python. This pre-processing ensures that the user input adheres to the required sentence structures.

### 3.2 PARSING MULTIPLE NAMES IN A SENTENCE

A dedicated method to parse multiple names called `parse_names()` was implemented to allow the chatbot to parse at least two or more names at once. This is made possible by looping through the split words in the input string and identifying unique words. The loop checks for two things — First, it checks if the word is not a “non-name word” [a, an, the, and, is, are, of]. Second, it checks if the word is not a “relation word” [parent, father, mother, ... ]. If these two conditions are satisfied, the word is recognized as a name and subsequently appended to the list of names. Finally, it returns the parsed names.

### 3.3 HOW THE CHATBOT HANDLES QUESTIONS

If a question mark ‘?’ is present at the end of the input sentence, it is recognized as a question and is processed through the ‘`process_question()`’ method. Based on the given sentence structures, the first word of a question string is always understood to be the question word [Are, Is, Who]. After identifying the question word, the names in the sentence are extracted through the `parse_names()` method and produces the appropriate output depending on the type of question being asked. A ‘Who’ question will display a list of names associated with the relation and name being referred to. However, if the relation prompted by the user yields no results, it will tell the user that no such family relation exists for that person. On the other hand, ‘Is’ and ‘Are’ questions will output ‘Yes’ or ‘No’ depending on the result of the Prolog query.

### 3.4 HOW THE CHATBOT HANDLES STATEMENTS

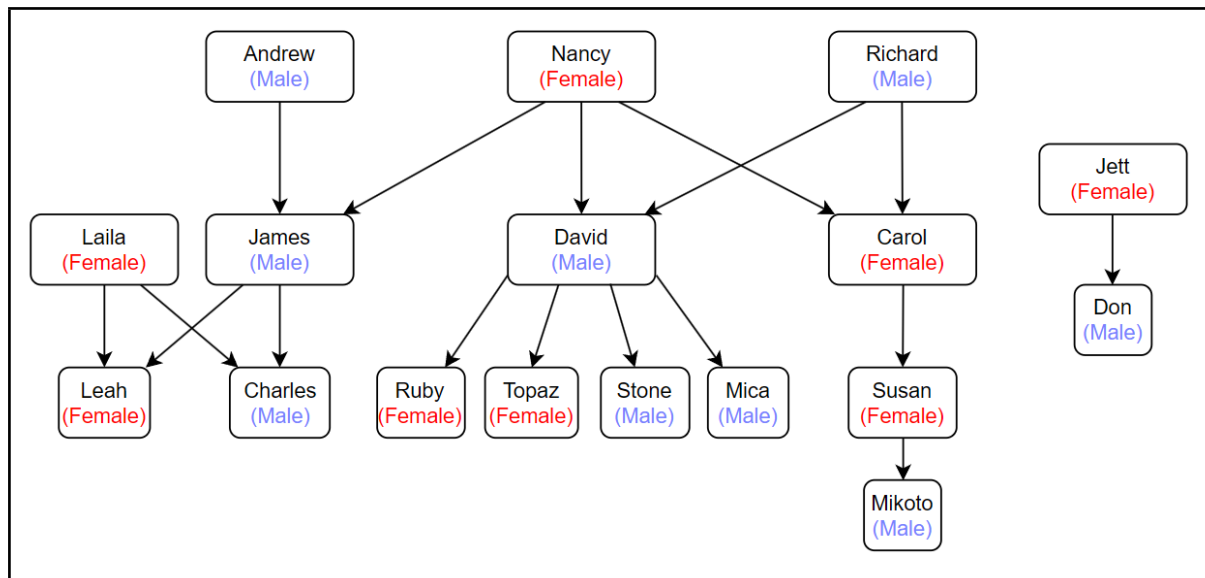
In contrast, if a period ‘.’ is present in the input sentence, it is recognized as a statement and is processed with the help of the ‘`process_statement()`’ method. If the queried relationship implied in the statement returns true, entailment occurs and the bot replies “I already knew that!”. Otherwise, the query returns false indicating that either a contingency or contradiction has taken place — the relationship can be added as new information or is deemed impossible to happen.

Depending on the relation specified in the statement, the bot processes whether the false statement is a contingency or contradiction by querying the rule “`check_{relation}`” from the knowledge base depending on the relation specified. These conditional checks determine whether it is feasible to input the new relationship into the database based on the existing knowledge of the bot. If the check returns false, a contradiction occurs as the relationship is impossible and cannot be added.

On the other hand, false statements caused by a contingency will return true for “`check_{relation}`” as this implies that the relationship between the people specified is feasible, the bot simply has no knowledge of it yet. The chatbot takes in the new knowledge through the Prolog meta-predicate `assert`. With the new information, the bot is able to infer other possible facts based on currently existing facts in the knowledge base.

## IV. RESULTS

The chatbot exhibits intelligent behaviour and is capable of making inferences on statements despite the relationship not being explicitly stated by the user. To showcase the bot in action, we encode the following family tree into the database:



Some scenarios that depict the intelligent behaviour of the chatbot are as follows. Take note that each code block represents a different program run and are not to be associated with one another:

### Addressing the issue of half-siblings

```
Ask or tell the bot something! Otherwise, say 'bye' to exit:
>>> Who are the siblings of James?
David
Carol
```

According to the current family tree, James, David and Carol are considered half-siblings as they share the same mother (Nancy), but have different fathers. Due to the possibility of half-siblings occurring in the database, asserting two people to be siblings does not automatically imply they share the same parents. So, if a new person is asserted as a sibling of someone with an existing parent, the parents of the new sibling will be unknown until new information is fed into the database.

```
Ask or tell the bot something! Otherwise, say 'bye' to exit:
>>> Don and Mordecai are siblings.
OK! I learned something.

Ask or tell the bot something! Otherwise, say 'bye' to exit:
>>> Who are the parents of Don?
Jett

Ask or tell the bot something! Otherwise, say 'bye' to exit:
>>> Who are the parents of Mordecai?
No parent found!
```

### Checking if two people are relatives

```
Ask or tell the bot something! Otherwise, say 'bye' to exit:
>>> Are Jett and Susan relatives?
No!

Ask or tell the bot something! Otherwise, say 'bye' to exit:
>>> Are Andrew and Mikoto relatives?
Yes!
```

A person is considered a relative of another as long as they are connected in some way in the family tree. Jett is unconnected from the main family tree, so she is not considered a relative of Susan and vice versa. On the other hand, Mikoto and Andrew are relatives because they are connected within the same family tree, Andrew being one of the spouses of Mikoto's great-grandmother, Nancy.

### **4.1 INSERTING NEW RELATIONSHIPS INTO THE KNOWLEDGE BASE**

The knowledge also facilitates the adding of new information into the database through statements. Below we see the relationships of new people being added, and some basic cases of intelligent inference.

```
Ask or tell the bot something! Otherwise, say 'bye' to exit:
>>> One is the father of Two.
OK! I learned something.

Ask or tell the bot something! Otherwise, say 'bye' to exit:
>>> Two is the father of Three.
OK! I learned something.

A person cannot be the father of its ancestor
Ask or tell the bot something! Otherwise, say 'bye' to exit:
>>> Three is the father of One.
That's impossible!

A grandfather cannot be the father of its grandchild
Ask or tell the bot something! Otherwise, say 'bye' to exit:
>>> One and Two are the parents of Three.
That's impossible!

One is unconnected from both Ruby's and Jett's family trees
Ask or tell the bot something! Otherwise, say 'bye' to exit:
>>> Are One and Ruby relatives?
No!

Ask or tell the bot something! Otherwise, say 'bye' to exit:
>>> Are One and Jett relatives?
No!
```

### Determining the gender of a person

When adding a new person to the database, asserting them with a gender-neutral term such as siblings, parents and children implies that their gender will be unknown until the bot feeds new data into the knowledge base as shown below.

Ask or tell the bot something! Otherwise, say 'bye' to exit:

>>> Ben and Richard are siblings.

OK! I learned something.

Ask or tell the bot something! Otherwise, say 'bye' to exit:

>>> Richard is the father of Peter.

OK! I learned something.

**The bot answers "No" because Ben's gender is currently unknown**

Ask or tell the bot something! Otherwise, say 'bye' to exit:

>>> Is Ben an uncle of Peter?

No!

**Ben cannot be a father of his sibling**

Ask or tell the bot something! Otherwise, say 'bye' to exit:

>>> Ben is a father of Richard.

That's impossible!

**Ben is finally determined to be a male**

Ask or tell the bot something! Otherwise, say 'bye' to exit:

>>> Ben is a brother of Richard.

OK! I learned something.

**The bot is able to infer Peter's uncle Ben without being explicitly told about it**

Ask or tell the bot something! Otherwise, say 'bye' to exit:

>>> Is Ben an uncle of Peter?

Yes!

### Attempting to assert someone as a grandfather of themselves

Ask or tell the bot something! Otherwise, say 'bye' to exit:

>>> Hades is the father of Zagreus.

OK! I learned something.

**A person cannot be a grandfather of themselves**

Ask or tell the bot something! Otherwise, say 'bye' to exit:

>>> Hades is the grandfather of Hades.

That's impossible!

**Hades is already a father of Zagreus, he cannot be Zagreus' grandfather**

Ask or tell the bot something! Otherwise, say 'bye' to exit:

>>> Hades is the grandfather of Zagreus.

That's impossible!

### Attempting to assert the wrong gendered term to someone

Ask or tell the bot something! Otherwise, say 'bye' to exit:

>>> Mark is the father of Patricia.

OK! I learned something.

Not possible to assert Mark as daughter as it was established that he is a male based on the previous assert

Ask or tell the bot something! Otherwise, say 'bye' to exit:

>>> Mark is a daughter of Ann.

That's impossible!

Ask or tell the bot something! Otherwise, say 'bye' to exit:

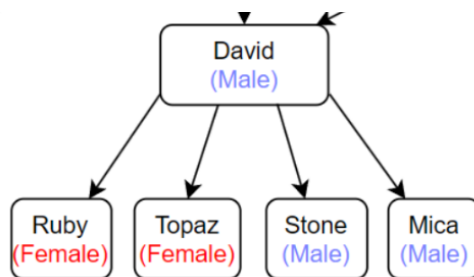
>>> Mark is a son of Ann.

OK! I learned something.

## 4.2 APPENDING NEW RELATIONSHIPS TO THE EXISTING FAMILY TREE

This section delves into more specific scenarios that involve the creation of new relationships to be added to the existing family tree, rather than stand-alone relationships.

### Adding new children under David



Ask or tell the bot something! Otherwise, say 'bye' to exit:

>>> Who are the children of David?

Mica

Topaz

Ruby

Stone

Ask or tell the bot something! Otherwise, say 'bye' to exit:

>>> Ruby, Topaz, Stone and Mica are children of David.

I already know that!

Andrea and Lorax don't currently exist in the database, so the bot learned something new. It also supports the processing of  $\geq 3$  children in one statement as seen below

Ask or tell the bot something! Otherwise, say 'bye' to exit:

>>> Ruby, Topaz, Stone, Mica, Andrea and Lorax are children of David.

OK! I learned something.

Ask or tell the bot something! Otherwise, say 'bye' to exit:

>>> Who are the children of David?

Lorax ☐ new

Ruby

Mica

Andrea ☐ new

Topaz

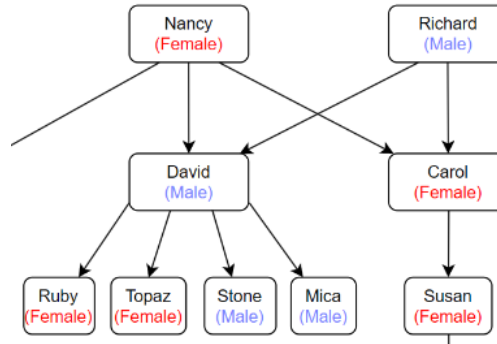
Stone

Ruby, Lorax and Andrea share a parent David, so they are considered siblings

Ask or tell the bot something! Otherwise, say 'bye' to exit:

```
>>> Who are the siblings of Ruby?
Lorax ☐ new
Mica
Andrea ☐ new
Topaz
Stone
```

### Adding Yvonne as a mother of Ruby, and Yvonne's parents



Ask or tell the bot something! Otherwise, say 'bye' to exit:

```
>>> Yvonne is the mother of Ruby.
```

OK! I learned something.

Ask or tell the bot something! Otherwise, say 'bye' to exit:

```
>>> Who are the parents of Ruby?
```

Yvonne ☐ new

David

Ask or tell the bot something! Otherwise, say 'bye' to exit:

```
>>> Allie and Morrie are the parents of Yvonne.
```

OK! I learned something.

**Allie's gender is determined through this statement**

Ask or tell the bot something! Otherwise, say 'bye' to exit:

```
>>> Allie is a grandmother of Ruby.
```

OK! I learned something.

**There cannot be another grandparent for Ruby as a child has at the maximum, four grandparents. Currently, Ruby's grandparents are Allie, Morrie, Nancy and Richard**

Ask or tell the bot something! Otherwise, say 'bye' to exit:

```
>>> Omar is a grandfather of Ruby.
```

That's impossible!

**The bot learned the gender of Yvonne's parent Morrie**

Ask or tell the bot something! Otherwise, say 'bye' to exit:

```
>>> Morrie is the father of Yvonne.
```

OK! I learned something.

**Ruby now has two grandfathers and two grandmothers.**

Ask or tell the bot something! Otherwise, say 'bye' to exit:

```
>>> Is Morrie a grandfather of Ruby?
```

Yes!

Ask or tell the bot something! Otherwise, say 'bye' to exit:

>>> Is Richard a grandfather of Ruby?  
Yes!

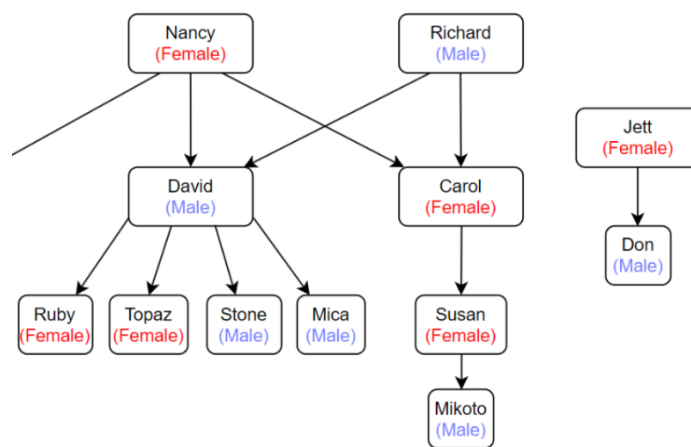
Ask or tell the bot something! Otherwise, say 'bye' to exit:

>>> Is Nancy a grandmother of Ruby?  
Yes!

Ask or tell the bot something! Otherwise, say 'bye' to exit:

>>> Is Allie a grandmother of Ruby?  
Yes!

### Adding Mordecai as a sibling of Don, and appending the disconnected tree to the main family tree



Currently, the tree is disconnected from the main tree

Ask or tell the bot something! Otherwise, say 'bye' to exit:

>>> Are Jett and Mikoto relatives?  
No!

Ask or tell the bot something! Otherwise, say 'bye' to exit:

>>> Don and Mordecai are siblings.  
OK! I learned something.

Merge the trees together by having Jett become a child of Ruby

Ask or tell the bot something! Otherwise, say 'bye' to exit:

>>> Ruby is the mother of Jett.  
OK! I learned something.

The new sibling Mordecai is consequently recognized as a relative of Nancy after connecting the trees

Ask or tell the bot something! Otherwise, say 'bye' to exit:

>>> Are Nancy and Mordecai relatives?  
Yes!

Richard is an ancestor of Mordecai, so this cannot be the case

Ask or tell the bot something! Otherwise, say 'bye' to exit:

>>> Mordecai is the father of Richard.  
That's impossible!



## V. LIMITATIONS/CHALLENGES

Throughout the development of the ChatBot, defining the rules in the knowledge base to allow for intelligent inferencing was found to be particularly challenging. As seen previously, the Knowledge Base included relationships that went beyond the surface of what was demonstrated in the specifications. Moreover, identifying whether a statement was a contingency or contradiction was a tricky process. Formulating the logic to prevent the asserts of invalid statements with the “check\_{relation}” rules proved to be rather difficult initially. Out of all the challenges faced, it was testing and debugging the bot that proved to be most cumbersome. Taking into account every possible real-world scenario and translating them into a logic-based inference system was no trivial task.

Comparing this familial relationship Chatbot with existing LLM based chatbots like ChatGPT, the latter already outperforms in terms of sentence parsing. ChatGPT is able to grasp the intentions of the user more deftly than this chatbot which is limited to being able to process specific sentence structures only. Apart from that, a limitation of this chatbot is its possibly slow processing time, which is apparent when asking if two people are relatives. This is due to the recursive nature of some facts encoded in the knowledge base which is present in the ancestor(X,Y) rule. Another factor that may contribute to the possible slow processing time is when there are already several facts in the knowledge base as a result of asserting new facts through statements.

## VI. CONCLUSION/REFLECTION

In summary, the group was able to come up with functions that parses the given statement or question in a way it can be able to extract names and relation words such as ‘child’, ‘father’, ‘aunt’ or ‘sibling’ so that the chatbot can be able to process and answer the user with the extracted words. The bot also looks for certain words or characters such as ‘are’, ‘is’, ‘who’ and ‘?’ to determine if the input was a question or a statement. With the help of the rules and predicates found in the knowledge base and the extracted words the process functions can be able to answer the user accordingly.

The project gave the researchers first hand experience in creating a logic-based inference system for the chatbot through Prolog. The group was also able to delve into rule creation for the familial relationships and think of ways on how the chatbot will be able to process it. The project made us think fundamentally about what makes a person a father, an uncle, etc.. Overall the project gave the researchers valuable insight on how these chatbots and other systems alike work. It showed its implications and promising potential in the future as it continues to develop and advance.

## CONTRIBUTIONS

Name of Member	Tasks
----------------	-------

1. Gutierrez, Allen Andrei D	Knowledge base, Limitations/Challenges, Introduction
2. Palacios, Jeremy Koen K.	Knowledge base, Conclusion/Reflection, Limitations/Challenges
3. Sabater, Cassandra Jo T.	Code, Knowledge base, Chatbot implementation, Result, Limitations/Challenges
4. Thay, Jerrel Chadler O.	Introduction, Knowledge base, Conclusion/Reflection

## REFERENCES

Banas, D. (2015, August 15). Prolog Tutorial [Video]. YouTube.  
<https://www.youtube.com/watch?v=SykxWpFwMGs>

Breuss, M. (2022, October 12). ChatterBot: Build a chatbot with Python – Real Python.  
 Python Tutorials – Real Python.  
<https://realpython.com/build-a-chatbot-python-chatterbot/>

Prolog - Relations. (n.d.). Online Tutorials, Courses, and eBooks Library | Tutorialspoint.  
[https://www.tutorialspoint.com/prolog/prolog\\_relations.htm](https://www.tutorialspoint.com/prolog/prolog_relations.htm)

Prolog recursion. (2023, April 6). EDUCBA. <https://www.educba.com/prolog-recursion/>