MCO REPORT

Introduction to Intelligent Systems

TERM 1 AY 2023 - 2024

S11 - Group 9

Submitted by:

Alabat, Jeanne Hazel

Garcia, Reina Althea

Jim, Mark Edison

Meneses, Hans Christian

Valdez, Katarina Milcah

Submitted to:

Zamin, Norshuhani

Date:

11/28/23

Table of Contents

I. Brief Introduction:

This chatbot is developed with a capability to understand and respond to specific statements or question patterns related to family relationships using a structured set of facts and rules. Users can provide facts through statements and the chatbot will provide relevant responses based on the questions, which will use the predefined rules and data it has.

II. Knowledge Base:

The knowledge base implemented in Prolog uses Pyswip to be accessed by python and assert facts and make queries.

Assertion:

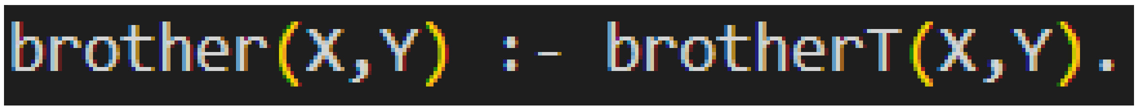
To prevent having stack overflows, having unique predicate names was crucial, as it tends to keep on checking rules and never arrive at a conclusion. When asserting a fact into the knowledge base it goes through 3 types of rules, the first is its own predicate name which ends with a capital T, this checks whether a rule already exists. The second rule is about “not” rules which regulates the inserted facts and rejects asserted facts when rules are broken. Finally the third rule is about the specific family relation, this is used when a query is made, or when a rule is asserted. These rules are used to expand more rules from the facts that were fed into the knowledge base.

Queries:

When asking a query the knowledge base uses the normal naming convention. This is because it uses backward chaining on facts that are not yet readily available on the knowledge base, it will use the rules given to come up with an answer.

Assert Rules:

When asserting a fact we used predicates with a capital T at the end of the predicate name in order to identify it as an assert. This is to prevent the chatbot from backward chaining and causing an infinite loop, when asserting a fact to the knowledge base. The cause of the infinite loop was due to predicates having the same name, and Prolog keeps on repeating the facts until the program crashes. To prevent this, using unique predicate names was the solution.



```
brother(X,Y) :- brotherT(X,Y).
```

Translation to First Order Logic Notation:

- $\forall X \; \forall Y \; (\text{brotherT}(X,Y) \rightarrow \text{brother}(X,Y))$

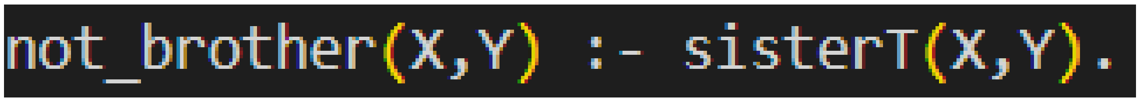
  This rule converts the asserted fact into a fact, which will be used by the knowledge base for queries and knowledge expansion.

- $\forall X \; \forall Y \; ((\text{sisterT}(X,Y) \lor \text{brotherT}(X,Y)) \rightarrow \text{siblingsT}(X,Y))$

  When a fact is true these facts are expanded to new facts, which can lead to new knowledge for the knowledge base.

Not Rules:

When asserting a rule it goes through its corresponding “not” rule to assess whether the fact to be asserted is valid or not. When a “not” fact becomes true, it will reject the facts and not assert it.



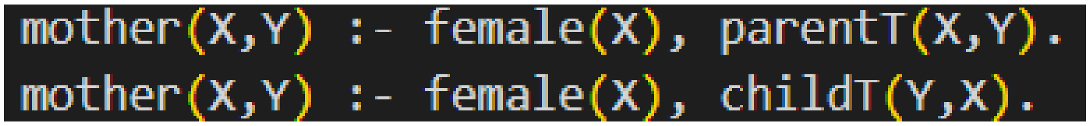
```
not_brother(X,Y) :- sisterT(X,Y).
```

Translation to First Order Logic Notation:

1. Rules violate family relationship generations older or younger than the other
   - $\forall X\ \forall Y$ (isOlder(X,Y) $\vee$ isYounger(X,Y) → not_siblings(X,Y))
   - grandparent(Y,X) → not_child(X,Y))
2. Person swapping gender
   - $\forall X\ \forall Y$ (male(X) $\vee$ isOlder(X,Y)) → not_daughter(X,Y))
3. If X and Y are the same person
   - $\forall X \forall Y$ ((X=Y) → not_sister(X,Y))
4. If X is a parent or not siblings of Y’s parent, it will result to not uncles or aunt
   - $\forall X\ \forall Y$ (male(X) $\vee$ (parent(Z,Y) $\wedge$ not_siblings(X,Z)) → not_aunt(X,Y))
5. If roles were swapped between parent and child relationships
   - $\forall X\ \forall Y$ (isOlder(Y,X) $\vee$ isYounger(X,Y) $\vee$ siblingsT(X,Y) → not_parent(X,Y))
6. If there is an existing father/mother for the child
   - $\forall X\ \forall Y$ (father(_Z,Y) → not_father(X,Y))

Family Specific Rules:

These are the rules that accept the Asserted rules and expand facts known into the knowledge base.



```
mother(X,Y) :- female(X), parentT(X,Y).
mother(X,Y) :- female(X), childT(Y,X).
```

Translation to First Order Logic Notation:

General Rule When Expanding Knowledge:

- ∀X ∀Y¬(X=Y)

  A person cannot have a relationship with themselves.

Relatives

- ∀X ∀Y (Relatives(X, Y) ↔ Grandparent(X, Y) ∨ Auntle(X, Y) ∨ Parent(X, Y) ∨ Child(X, Y) ∨ Siblings(X, Y))

  The relatives relationship considers every other relationship, as long as X and Y are connected in some way based on the list of relationships.

Brother

- ∀X ∀Y (male(X) ∧ siblingsT(X,Y)→brother(X,Y))
- ∀X ∀Y (uncleT(X,Z) ∧ parentT(Y,Z) → brotherT(X,Y))

  The brother relationship is considered through the fact that the person is male and siblings, and the uncle and parent or child relationship where the uncle of a child is a brother of a parent of the same child.

Sister

- $\forall X \ \forall Y$ (female(X) $\wedge$ siblingsT(X,Y) → sister(X,Y))
- $\forall X \ \forall Y$ (auntT(X,Z) $\wedge$ childT(Z,Y) → sisterT(X,Y))

The sister relationship considers the fact sisterT, infers through female gender and siblings, and the aunt and parent or child relationship similar to the rules of brother.

Siblings

- $\forall X \ \forall Y$ ((brother(X,Y) $\vee$ sister(X,Y)) → siblings(X,Y))
- $\forall X \ \forall Y$ (siblingsT(X,Y) $\vee$ siblingsT(Y,X) → siblings(X,Y))
- $\forall X \ \forall Y$ (parent(Z,X) $\wedge$ parent(Z,Y) → siblings(X,Y))

The sibling relationship encompasses many relationships. It includes the brother or sister, a direct relationship of siblings where it considers siblings of a sibling, children of the same parent and its child counterpart.

Uncle

- $\forall X \ \forall Y$ (brother(X,Z) $\wedge$ parentT(Z,Y) → uncle(X,Y))
- $\forall X \ \forall Y$ (brother(X,Z) $\wedge$ childT(Y,Z) → uncle(X,Y))

The uncle relationship considers the basic rule that if X is a brother of a parent who has a child Y, then X is an uncle of Y.

Aunt

- $\forall X \ \forall Y$ (sister(X,Z) $\wedge$ parentT(Z,Y) $\rightarrow$ aunt(X,Y))
- $\forall X \ \forall Y$ (sister(X,Z) $\wedge$ childT(Y,Z) $\rightarrow$ aunt(X,Y))

This is the counterpart of Uncle and considers sister instead of brother.

Parent

- $\forall X \ \forall Y$ (childT(Y,X) $\rightarrow$ parent(X,Y))
- $\forall X \ \forall Y$ (parentT(X,Z) $\wedge$ siblingsT(Y,Z) $\rightarrow$ parent(X,Y))
- $\forall X \ \forall Y$ (father(X,Y) $\vee$ mother(X,Y) $\rightarrow$ parent(X,Y))

The parent relationship considers the parentT fact, its counterpart relationship Child, siblings of the same parent, the father and mother relationship, and the grandparent child relationship where if X is a grandparent of Z and Y is a parent of Z, then X is a parent of Y.

Father

- $\forall X \ \forall Y$ (male(X) $\wedge$ parentT(X,Y) $\rightarrow$ father(X,Y))
- $\forall X \ \forall Y$ (male(X) $\wedge$ grandparent(X,Z) $\wedge$ childT(Z,Y) $\rightarrow$ father(X,Y))

The father relationship considers the fatherT fact, infers through the male gender and parent, if it is the father of a child with siblings, and infers through the male gender involving grandparent and parent relationships.

Mother

- $\forall X \; \forall Y$ (female(X) $\wedge$ childT(Y,X) $\rightarrow$ mother(X,Y))
- $\forall X \; \forall Y$ (female(X) $\wedge$ grandparent(X,Z) $\wedge$ parentT(Y,Z) $\rightarrow$ mother(X,Y)) ;

The mother relationship considers the motherT fact, infers through the female gender and parent, if it is the mother of a child with siblings, and also infers through the female gender involving grandparent and parent relationships.

Child

- $\forall X \; \forall Y$ (parentT(Y,X) $\rightarrow$ child(X,Y))
- $\forall X \; \forall Y$ (sonT(X,Y) $\vee$ daughterT(X,Y) $\rightarrow$ childT(X,Y))

The child relationship considers the childT fact, its counterpart parent relationship, the grandparent and parent relationships where if X has a grandparent who is a parent of Y, then X is a child of Y, and also the son and daughter relationships.

Son

- $\forall X \; \forall Y$ (male(X) $\wedge$ childT(X,Y) $\rightarrow$ son(X,Y))
- $\forall X \; \forall Y$ (male(X) $\wedge$ parentT(Y,X) $\rightarrow$ son(X,Y))
- $\forall X \; \forall Y$ (male(X) $\wedge$ grandparent(Z,X) $\wedge$ childT(Y,Z) $\rightarrow$ son(X,Y))

The son relationship considers the sonT fact, infers through the male gender and both child and parent relationships, as well if Y is a grandparent of Z and X is a parent of Z and X is male, then X is a son of Y.

Daughter

- $\forall X\ \forall Y\ (female(X) \wedge childT(X,Y) \rightarrow daughter(X,Y))$
- $\forall X\ \forall Y\ (female(X) \wedge parentT(Y,X) \rightarrow daughter(X,Y))$
- $\forall X\ \forall Y\ (female(X) \wedge grandparent(Y,Z) \wedge parentT(X,Z) \rightarrow daughter(X,Y))$

The daughter relationship considers the daughterT fact, infers through the female gender and both child and parent relationships, as well if Y is a grandparent of Z and X is a parent of Z and X is female, then X is a daughter of Y. It is the counterpart of the Son relationship.

Grandparent

$\forall X\ \forall Y\ (grandfather(X,Y) \vee grandmother(X,Y) \rightarrow grandparent(X,Y))$

This is simply a predicate for a non gender grandfather and grandmother.

Grandfather

- $\forall X\ \forall Y\ (male(X) \wedge parentT(X,Z) \wedge parentT(Z,Y) \rightarrow grandfather(X,Y))$
- $\forall X\ \forall Y\ (male(X) \wedge childT(Y,Z) \wedge childT(Z,X) \rightarrow grandfather(X,Y))$

The grandfather relationship considers the grandfatherT fact, and infers through the male gender and if X is a parent of Z and Z is a parent of Y then X is a grandfather of Y, as well as its counterparts involving parent and child.

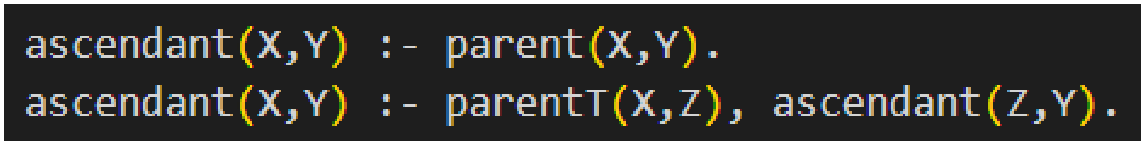
Grandmother

- ∀X ∀Y (female(X) ∧ parentT(X,Z) ∧ childT(Y,Z) → grandmother(X,Y))
- ∀X ∀Y (female(X) ∧ parentT(Z,Y) ∧ childT(Z,X) → grandmother(X,Y))

The grandmother relationship considers the grandmotherT fact, and infers through the male gender and if X is a parent of Z and Z is a parent of Y then X is a grandmother of Y, as well as its counterparts involving parent and child.

Recursive Rules:

These are the rules that use recursion to access the relationship between 3 or more people.



```
ascendant(X,Y) :- parent(X,Y).
ascendant(X,Y) :- parentT(X,Z), ascendant(Z,Y).
```

Translation to First Order Logic Notation:

Descendant

- ∀X ∀Y (child(X,Y) → descendant(X,Y))
- ∀X ∀Y (childT(X,Z) ∧ descendant(Z,Y) → descendant(X,Y))

The descendant relationship infers and recurses through the child relationship and is used for the isYounger predicate, for example, if we wish to know if X is a descendant of Y, we have to see if X is a child of someone younger than Y. It is the counterpart of the Ascendant relationship.

Ascendant:

- $\forall X \, \forall Y \, (\text{parent}(X,Y) \rightarrow \text{ascendant}(X,Y))$
- $\forall X \, \forall Y \, (\text{parentT}(X,Z) \land \text{ascendant}(Z,Y) \rightarrow \text{ascendant}(X,Y))$

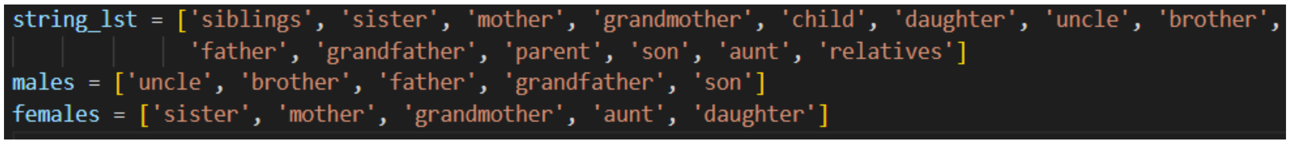
The ascendant relationship infers and recurses through the parent relationship and is used for the isOlder predicate, for example, if we wish to know if X is an ascendant of Y, we have to see if X is a parent of someone older than Y.

III. Chatbot Implementation:

The chatbot was implemented in python which uses basic sentence structure format to analyze the fact and query.

Detecting sentence structure:

The chatbot uses basic sentence patterns to differentiate queries from prompts, such as using the words “Is”, “Are”, and “Who” at the beginning of the sentence and using correct punctuations to let the bot know whether it is a fact or a query. Certain keywords for family relationships are stored in arrays for the chatbot to search in a sentence to know what the prompt or query will be about.



```
string_lst = ['siblings', 'sister', 'mother', 'grandmother', 'child', 'daughter', 'uncle', 'brother',
              'father', 'grandfather', 'parent', 'son', 'aunt', 'relatives']
males = ['uncle', 'brother', 'father', 'grandfather', 'son']
females = ['sister', 'mother', 'grandmother', 'aunt', 'daughter']
```

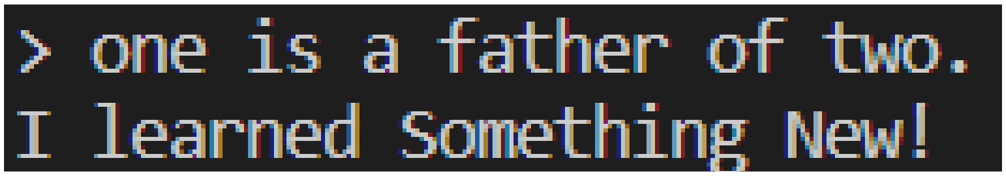
Asserting Facts:

For asserting facts it checks the structure of the input, when the last character of the string is a '.' then it is analyzed as asserting. It analyzes the names, when the name does not have any existing facts it accepts the asserted fact. When a fact is asserted it is a capital T on the end of the predicate, and it expands any facts it can produce with the given rules, using forward chaining. When the name has an existing fact, it assesses the knowledge base, whether a fact can be asserted or not.

Replies for Asserted Facts:

When Inserting a true fact:

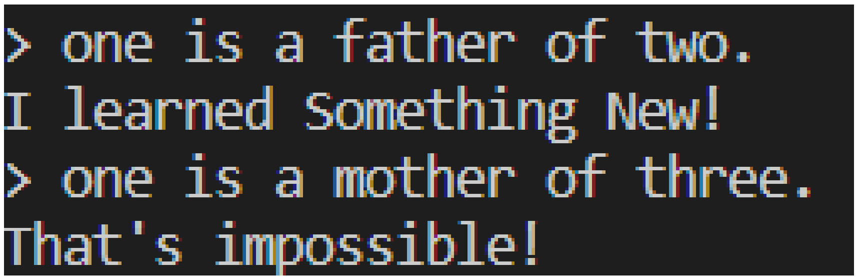
The fact is accepted into the knowledge base.



```
> one is a father of two.
I learned Something New!
```

When Inserting a false fact:

The chatbot will decline the asserted information, and will reply “That’s Impossible”.



```
> one is a father of two.
I learned Something New!
> one is a mother of three.
That's impossible!
```

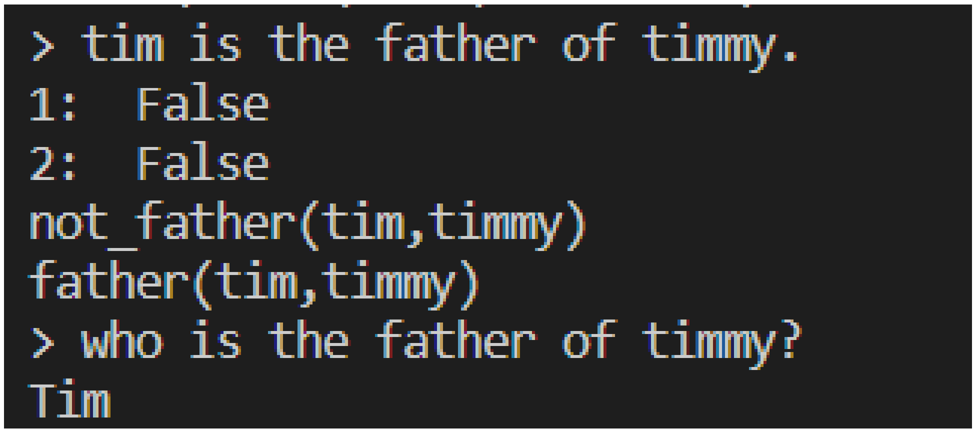
Making a Query:

For Queries the first pattern that it checks is the punctuation, if it ends with a question mark “?” then it is analyzed as a query. Then it categorizes the string into three types of sentences (Who, Is, Are), since there is a difference between the number of people a sentence structure handles. The first kind of query begins with the word “Who”, and these kinds of queries can only handle one person, which is always located at the end of the sentence before the punctuation. The second kind of query begins with the word “Is”, which always involves the relationship between two people. Finally the third kind of query begins with the word “Are” this kind of relationship always involves two or more people.

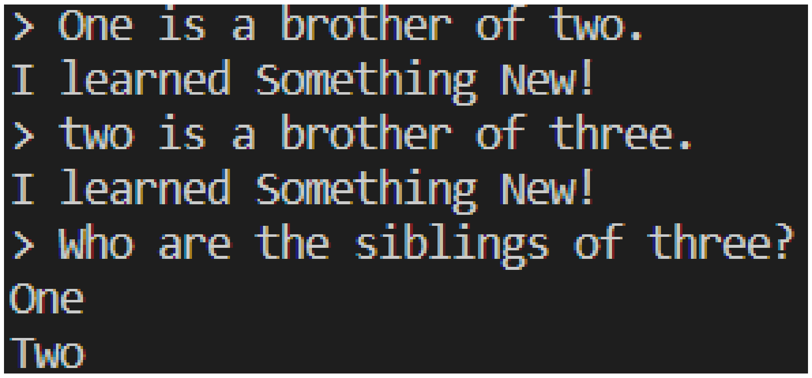
Replies for the Queries:

1. Who Queries

   This type of query always asks for a person/s with a specific relationship. The reply of the chatbot will be the name of the person/s in relation to the requested query.



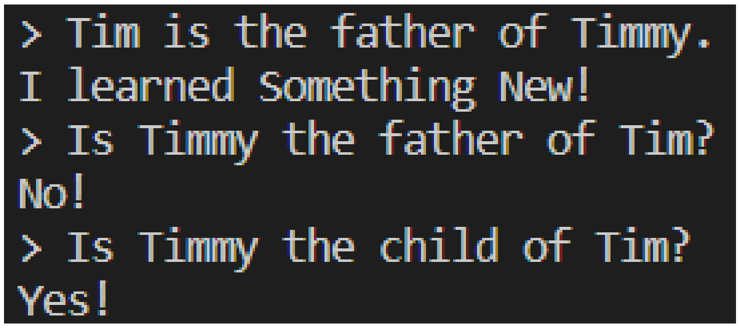
```
> tim is the father of timmy.
1:  False
2:  False
not_father(tim,timmy)
father(tim,timmy)
> who is the father of timmy?
Tim
```



```
> One is a brother of two.
I learned Something New!
> two is a brother of three.
I learned Something New!
> Who are the siblings of three?
One
Two
```

2. Is Queries

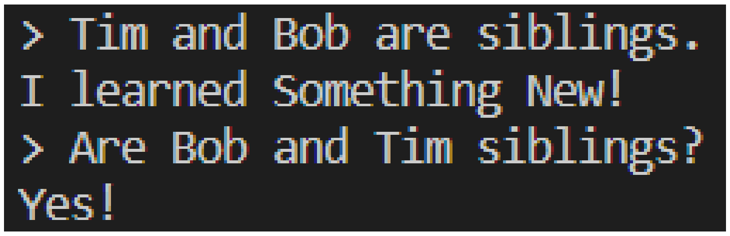
   In these types of queries, the query asks for the relationship between the two people involved. The reply of the chatbot would result in a yes or no.



```
> Tim is the father of Timmy.
I learned Something New!
> Is Timmy the father of Tim?
No!
> Is Timmy the child of Tim?
Yes!
```

3. Are Queries

These types of queries are similar to the 2nd type, but these queries can involve more than two people. The response of the chatbot would also result in a yes or no.



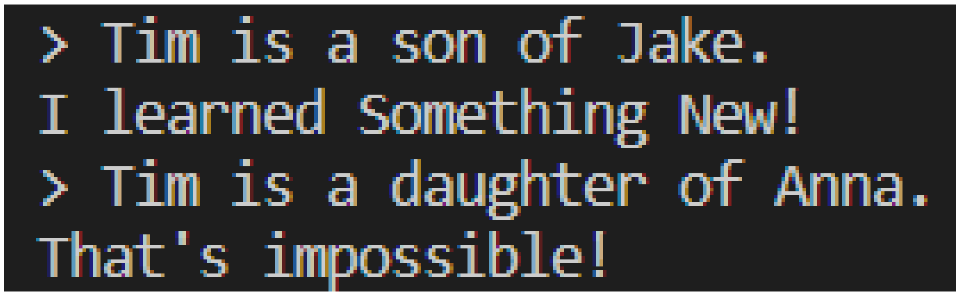
```
> Tim and Bob are siblings.
I learned Something New!
> Are Bob and Tim siblings?
Yes!
```

IV. Results:

The chatbot is capable of analyzing biological relationships, and is able to understand basic family relationships, and judge the facts it was fed and reply to the queries that were asked.

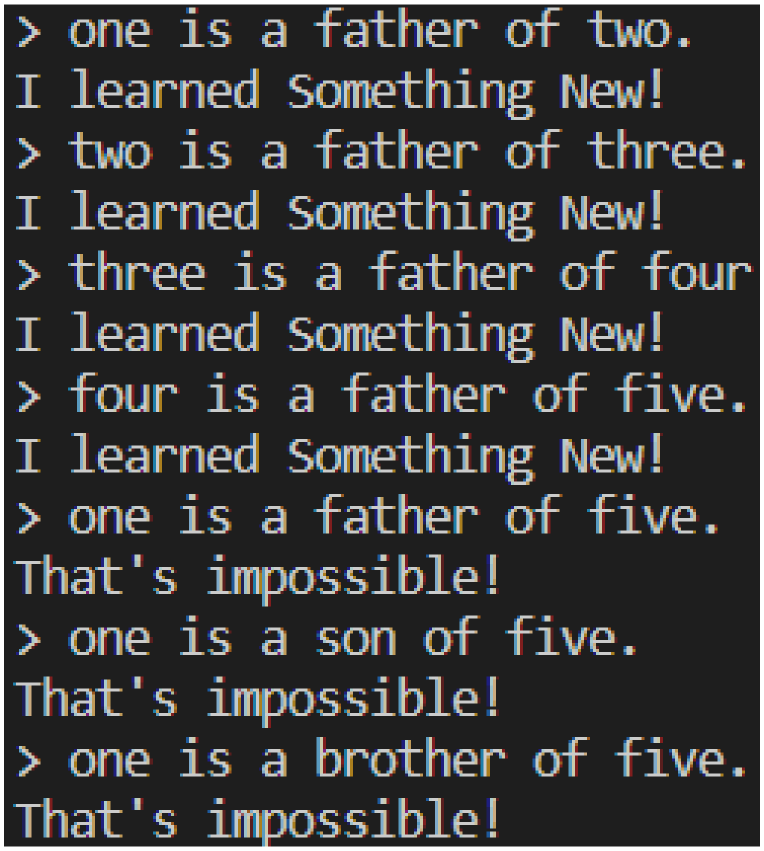
Capabilities:

1. It can analyze swapping of genders



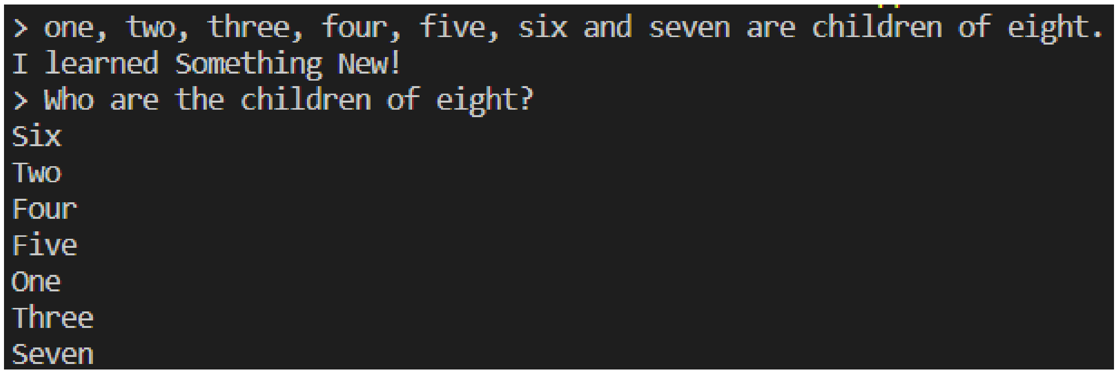
```
> Tim is a son of Jake.
I learned Something New!
> Tim is a daughter of Anna.
That's impossible!
```

2. It can reject facts regarding which generation they came from



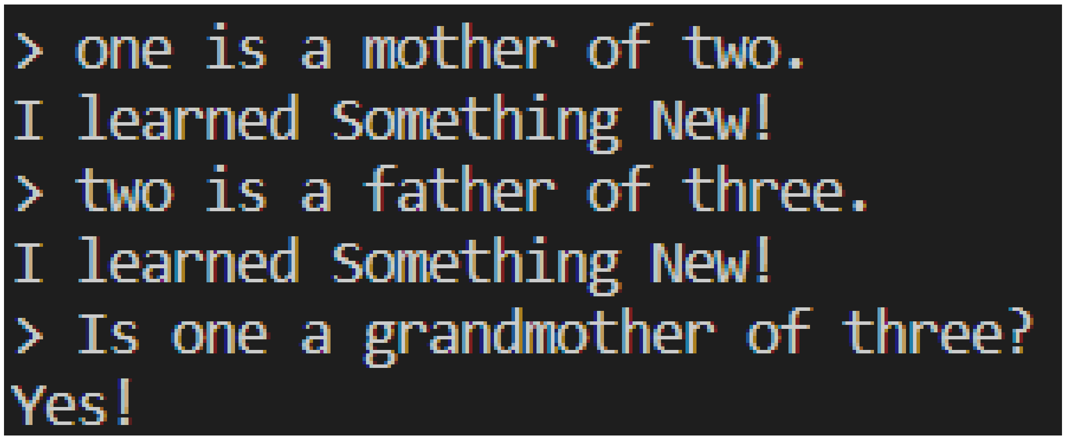
```
> one is a father of two.
I learned Something New!
> two is a father of three.
I learned Something New!
> three is a father of four
I learned Something New!
> four is a father of five.
I learned Something New!
> one is a father of five.
That's impossible!
> one is a son of five.
That's impossible!
> one is a brother of five.
That's impossible!
```

3. It can analyze multiple siblings relationships

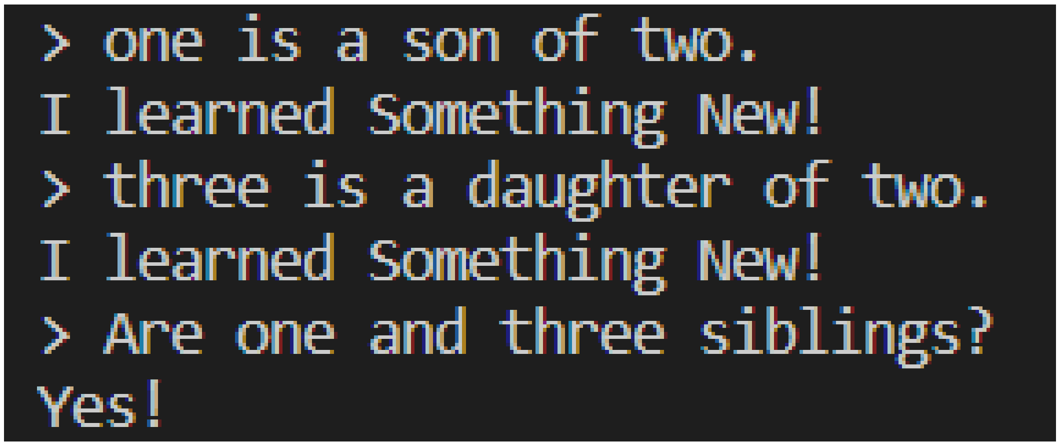


```
> one, two, three, four, five, six and seven are children of eight.
I learned Something New!
> Who are the children of eight?
Six
Two
Four
Five
One
Three
Seven
```

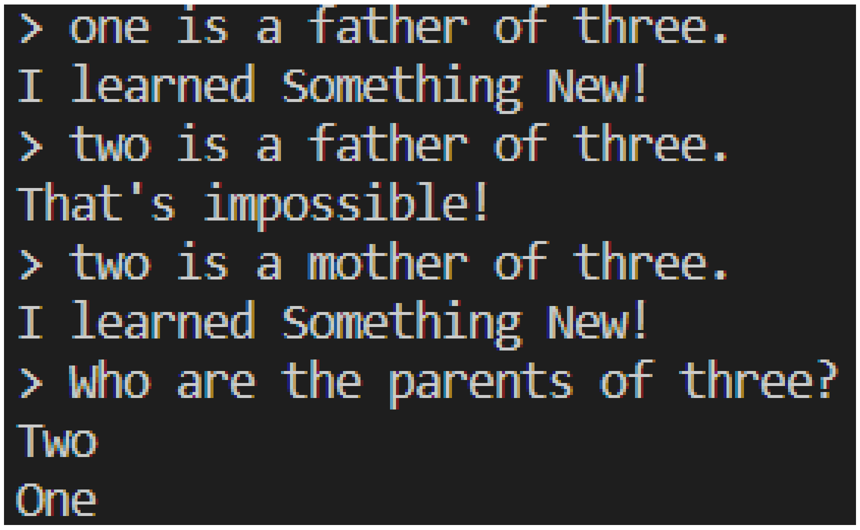
4. It can analyze grandfather/grandmother relationships (Note: grandparents are only gender specific, since there is no grandparent query in the specs)


```
> one is a mother of two.
I learned Something New!
> two is a father of three.
I learned Something New!
> Is one a grandmother of three?
Yes!
```

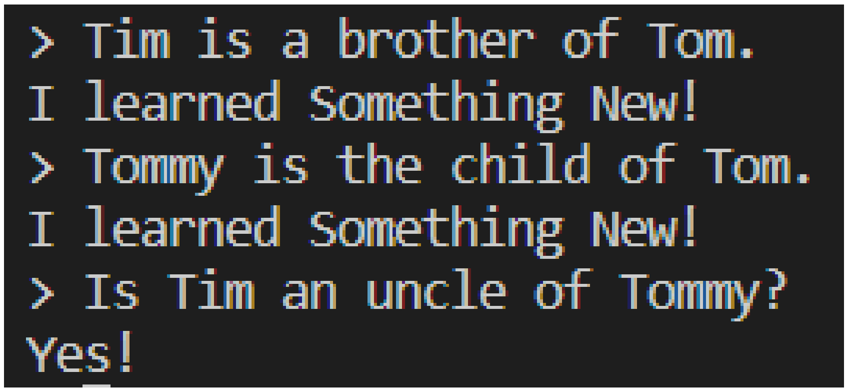
5. It can analyze sibling relationship through common parents


```
> one is a son of two.
I learned Something New!
> three is a daughter of two.
I learned Something New!
> Are one and three siblings?
Yes!
```

6. It can accept and reject parent child relationships

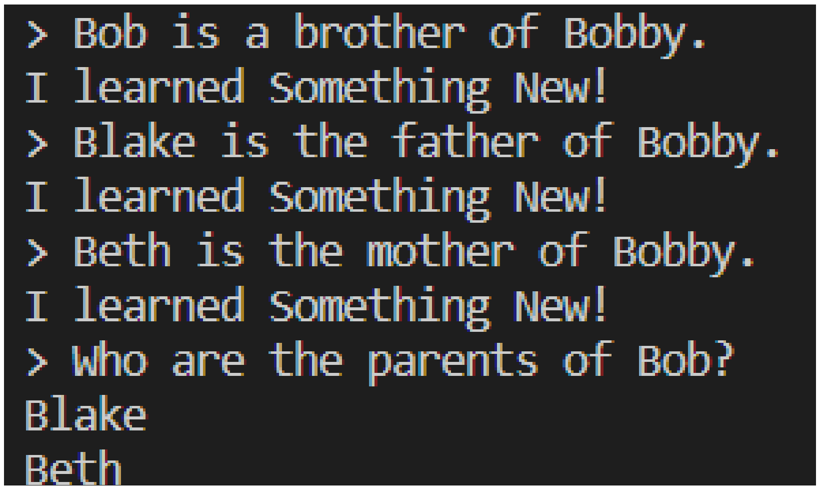

```
> one is a father of three.
I learned Something New!
> two is a father of three.
That's impossible!
> two is a mother of three.
I learned Something New!
> Who are the parents of three?
Two
One
```

7. It can analyze uncle and aunt relationships



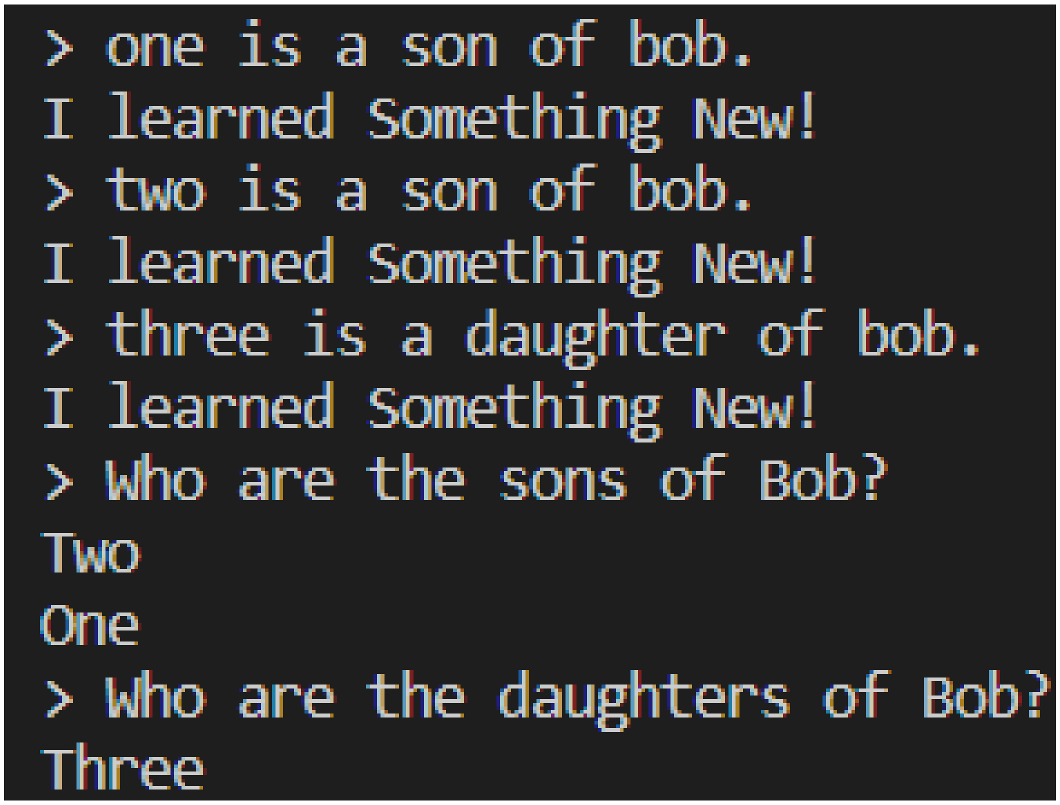
```
> Tim is a brother of Tom.
I learned Something New!
> Tommy is the child of Tom.
I learned Something New!
> Is Tim an uncle of Tommy?
Yes!
```

8. It is able to analyze the parents through sibling relationship



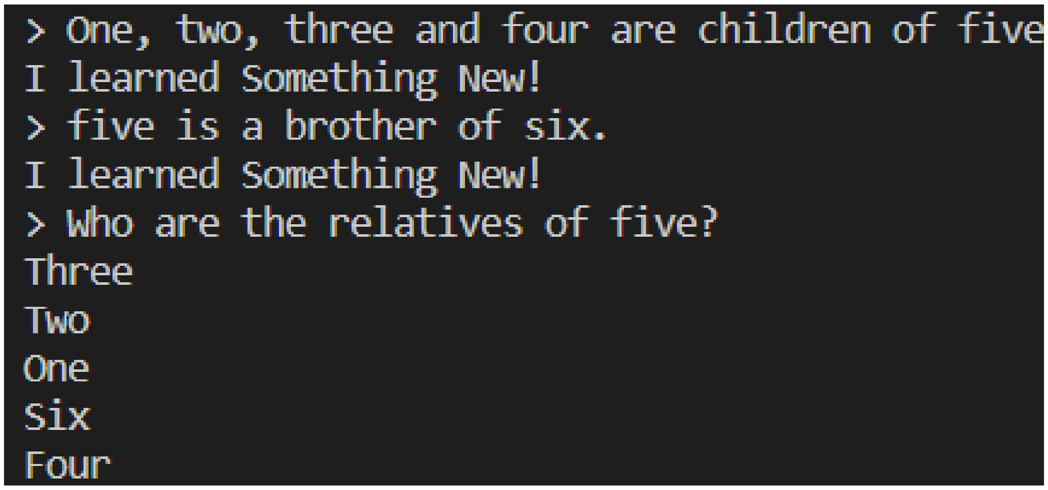
```
> Bob is a brother of Bobby.
I learned Something New!
> Blake is the father of Bobby.
I learned Something New!
> Beth is the mother of Bobby.
I learned Something New!
> Who are the parents of Bob?
Blake
Beth
```

9. It can distinguish sons and daughters



```
> one is a son of bob.
I learned Something New!
> two is a son of bob.
I learned Something New!
> three is a daughter of bob.
I learned Something New!
> Who are the sons of Bob?
Two
One
> Who are the daughters of Bob?
Three
```

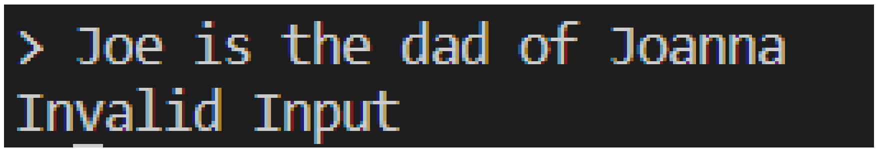
10. It can query relative/s



```
> One, two, three and four are children of five
I learned Something New!
> five is a brother of six.
I learned Something New!
> Who are the relatives of five?
Three
Two
One
Six
Four
```

Weaknesses:

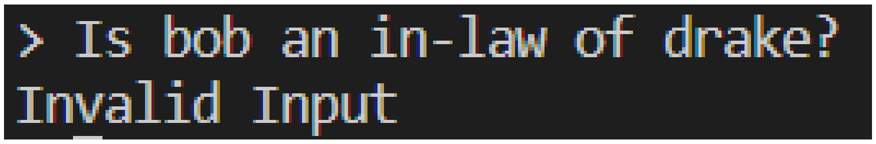
1. It has to follow specific sentence structure



```
> Joe is the dad of Joanna
Invalid Input
```

For the bot to understand an assertion or a query, it must use specific punctuation marks like commas, period, and question marks. The bot is also programmed to use specific names for relationships, so it cannot understand the words like mom and dad.

2. It does not query or understand in-law concepts



```
> Is bob an in-law of drake?
Invalid Input
```

Since the chatbot was limited to the specifications of the project, it does not accept queries, which involve in-laws and only accepts biological relationships.

3. The relationships it understands are limited to the specifications the project

   The project is only limited to the facts and queries which are linked to biological aspects such as mother, father, brother, sister, and more. This means that it cannot identify other relationships that are outside biological aspects, like marriage, friends, and acquaintances.

V. Limitations and Challenges:

Challenges:

One of the largest challenges tackled during the coding process was avoiding a stack overflow, such as repeatedly using same predicate names like siblings(X,Y) :- siblings(Y,X). Without tracing the prolog script would yield great chances of causing a stack overflow. A solution was to use unique predicate names, but it also lengthened the code significantly.

Another challenge tackled was having generations of 4 families. The solution to this problem was creating recursive code, for specific relationships, as there are specific relationships which would cause some problems. Before we used a pattern like parent(X,Z) parent(Z,Y), which can only cover 3 people in a specific relationship, which is why recursion was used in order to explore 4 or more generations of people.

Limitations:

In comparison to multiple Natural language processing softwares, our chatbot is only a portion of the capabilities of these softwares. One of the reasons is due to how it is created. Unlike our chatbot which is coded for its rules and syntax, LLM softwares like Chatgpt uses machine learning to learn given a large set of data. With this it improves as more unbiased and accurate data, unlike our chatbot which is limited to the given set of prompts, queries, and rules. Our chatbot would only improve, when more rules and code are applied to the program.

Another limitation for our chatbot would be it would not be able to make a reply, when no facts are asserted into the program, unlike Chatgpt which can give answers to queries without asking for information to the users. The chatbot is also limited to prompts using text, yet Chatgpt can accept images and analyze it.

During testing it was discovered that the chatbot tends to get slower, as the knowledge base gets larger. Some prompts would have delays in checking the facts and making a reply, as it cannot handle large amounts of data.

VI. Conclusion:

The students were able to create a chatbot, which was able to analyze family relationships given that facts were provided by the user. In order to avoid stack overflows during run time, using unique predicate names was beneficial. Making a large language model using this approach would take a large amount of time, code, and tracing before it can even parse prompts properly. The approach of making the chatbot was similar to a brute force approach, as every single case must be coded, compared to machine learning where the program learns the patterns of the data it was fed. Prolog was a powerful tool for creating chatbots for specific uses, but it has its limitations on how it improves, since it needs more rules in order to function properly.

VII. Table of Contributions:

Member	**Contribution**
Alabat, Jeanne Hazel	Documentation
Garcia, Reina Althea	Documentation
Jim, Mark Edison	Chatbot, Knowledge base, and Documentation
Meneses, Hans Christian	Knowledge base and Documentation
Valdez, Katarina Milcah	Documentation