King Fahd University of Petroleum & Minerals

College of Computing and Mathematics

Information & Computer Science

ICS 381: Principles of Artificial Intelligence

Assignment #5

Student Name: Abdullah Alzeid

Question 1:-

To determine if Tom has any grandchildren by using parent relation we need to define the following:

```
grandchild(Grandchild, Grandparent) :- % Question 1
        parent(Grandparent, Child),
        parent(Child, Grandchild).
```

Now we can get tom grandchildren by using the following query:

```
?- grandchild(Grandchild, tom).
Grandchild = ann ;
Grandchild = pat ;
Grandchild = jim ;
```

Question 2:-

Here are the results of the requested queries:

```
?- parent(jim,X).
false.

?- parent(X,jim).
X = bob.

?- parent(pam,X), parent(X,pat).
X = bob.

?- parent(pam,X), parent(X,Y), parent(Y,jim).
false.
```

Question 3:-

Who is Pat's parent ?

To find out who Pat's parent(s) are/is, we can use the query:

parent(X, pat).

Does Liz have a child ?

To determine if Liz has any children, we can use the query:

parent(liz, X).

Who is Pat's grandparent ?

To find out who Pat's grandparent(s) are/is, we can use the query :

parent(X, Y), parent(Y, pat).

```
?- parent(X, pat).
X = bob.

?- parent(liz, X).
false.

?- parent(X, Y), parent(Y, pat).
X = pam,
Y = bob ;
X = tom,
Y = bob ;
```

Question 4:-

To express “For all X and Y, if X is a parent of Y then Y is an offspring of X” we need to define the following:

```
offspring(Y, X) :- % Question 4
        parent(X, Y).
```

To demonstrate the rule above we can ask the following:

```
?- offspring(Child, Parent).
Child = bob,
Parent = pam ;
Child = bob,
Parent = tom ;
Child = liz,
Parent = tom ;
Child = ann,
Parent = bob ;
Child = pat,
Parent = bob ;
Child = jim,
Parent = bob.
```

Question 5:-

To express the grandparent relation we can define the following:

```
grandparent(Grandparent, Grandchild) :- % Question 5
        parent(Grandparent, Parent),
        parent(Parent, Grandchild).
```

And to demonstrate the rule above we can list all pairs of grandparents (G) and their grandchildren (GC) by using the following query:

```
?- grandparent(G, GC).
G = pam,
GC = ann ;
G = pam,
GC = pat ;
G = pam,
GC = jim ;
G = tom,
GC = ann ;
G = tom,
GC = pat ;
G = tom,
GC = jim ;
```

Question 6:-

To express the sister relation we can define the following rule:

```
sister(Sister, Person) :- % Question 6
        female(Sister),           % Sister is female
        parent(Parent, Sister),   % They share a parent
        parent(Parent, Person),   % Both are children of that parent
        Sister \= Person.         % And are not the same person
```

Then we can demonstrate the rule by listing all the sisters (if any) of the defined individual like the following:

```
?- sister(Sister, Person).
Sister = liz,
Person = bob ;
Sister = ann,
Person = pat ;
Sister = ann,
Person = jim ;
Sister = pat,
Person = ann ;
Sister = pat,
Person = jim.
```

Question 7:-

"Everybody who has a child is happy"

First, we introduce a one-argument relation happy which will be true if the person has at least one child.

```
happy(Person) :-
    parent(Person, _).
```

This rule states that a Person is happy if they are a parent of some child (denoted by _, which is a wildcard).

"For all X, if X has a child who has a sister then X has two children"

We can demonstrate this expression by defining the following:

```
has_two_children(Person) :-
    parent(Person, Child1),
    parent(Person, Child2),
    sister(Sister, Child1),
    Child1 \= Child2.
```

This rule states that Person has two children if Person is the parent of Child1 and Child2, and Child1 has a sister (which could be Child2 or another child). The condition Child1 \= Child2 ensures that Child1 and Child2 are not the same child, thus ensuring two distinct children.

We can demonstrate the first rule by listing all the individuals who are happy (have children) using the following query:

```
?- setof(Person, Child^happy(Person), HappyPeople).
HappyPeople = [bob, pam, tom].
```

In the above query we are using “setof” to collect all unique individuals who are happy instead of getting duplicate answers based on how many children each individual has.

Next to demonstrate the second rule we can list all individuals who have two children such that one of the children has a sister using the following query:

```
?- setof(Person, Child1^Child2^has_two_children(Person), PeopleWithTwoChildren).
PeopleWithTwoChildren = [bob, tom].
```

Again here we are using “setof” to get unique answers.

Question 8:-

To illustrate grandchild relation using parent relation we can define the following:

```
grandchild(Grandchild, Grandparent) :- % Question 8
        parent(Parent, Grandchild),
        parent(Grandparent, Parent).
```

We can demonstrate the above rule by listing all pairs of Grandchild and Grandparent for which the grandchild relationship holds true using the following query (zoom in to see the picture clearly):

setof((grandchild=Grandchild, grandparent=Grandparent), Parent^(parent(Parent, Grandchild), parent(Grandparent, Parent)), Result).

The output:

Result = [(grandchild=ann, grandparent=pam), (grandchild=ann, grandparent=tom), (grandchild=jim, grandparent=pam), (grandchild=jim, grandparent=tom), (grandchild=pat, grandparent=pam), (grandchild=pat, grandparent=tom)].

Question 9:-

To define the aunt relation we can use the previous “parent” and “sister” relations we already have defined to introduce the following:

```
aunt(X, Y) :- % Question 9
    parent(Parent, Y),
    sister(X, Parent).
```

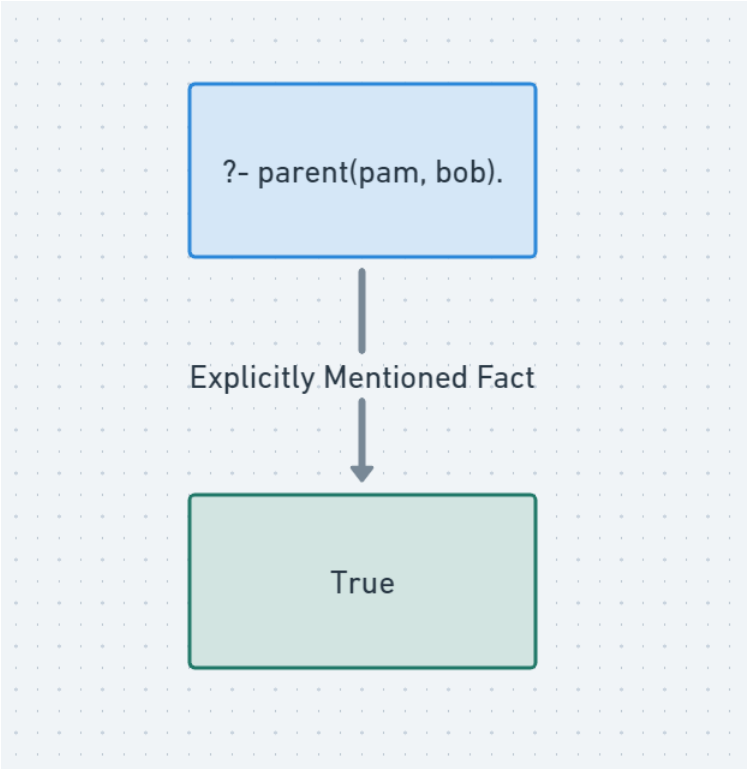
Then we can demonstrate the rule by listing all aunt-niece/nephew pairs using the following query:

```
?- aunt(Aunt, NieceNephew).
Aunt = liz,
NieceNephew = ann ;
Aunt = liz,
NieceNephew = pat ;
Aunt = liz,
NieceNephew = jim ;
```

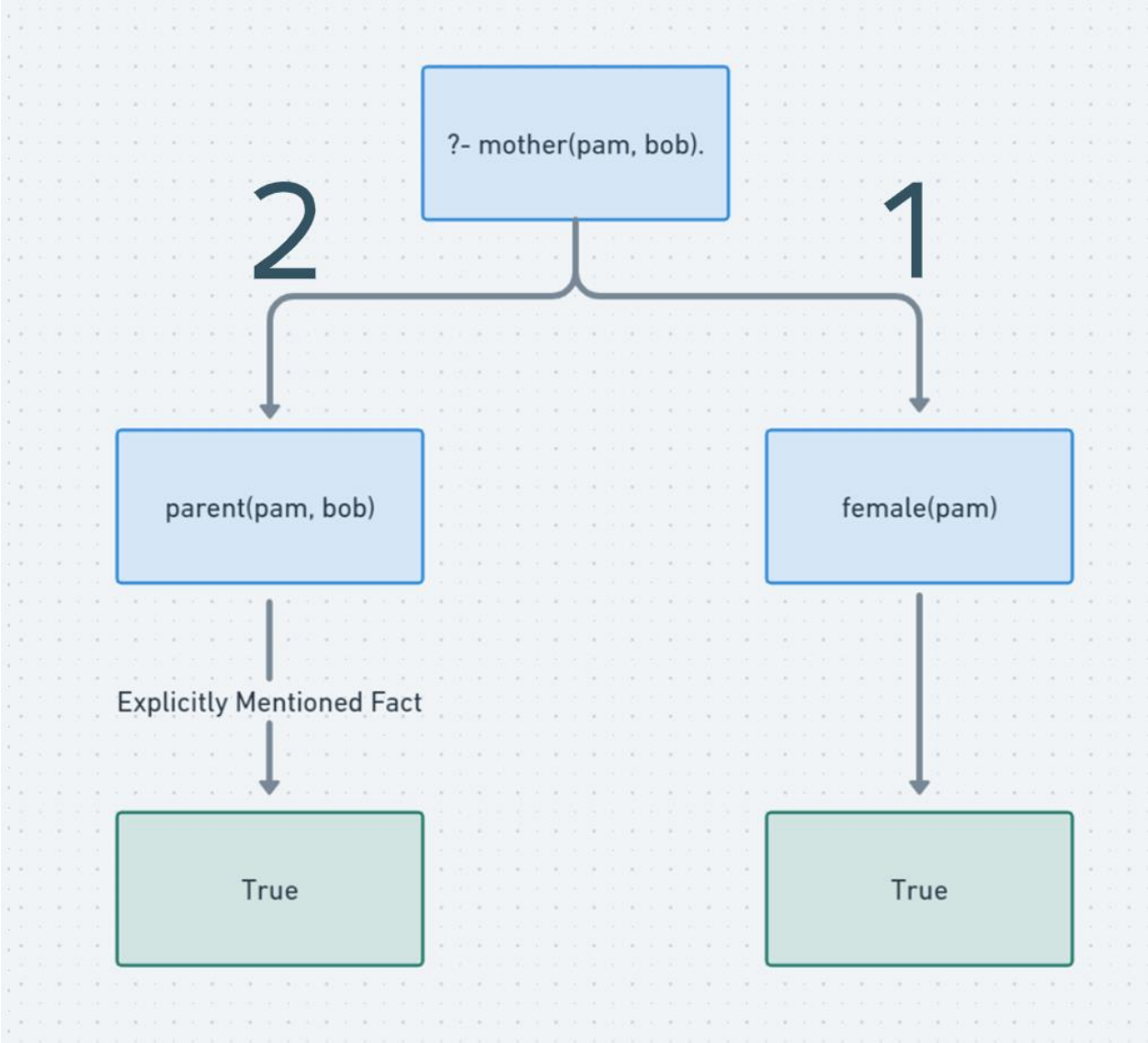
Note: For all the previous questions where we demonstrated any rule by listing all possible pairs of whatever was the relationship we are looking for. The rules can also be used to query the relationship for an individual person.

Question 10:-

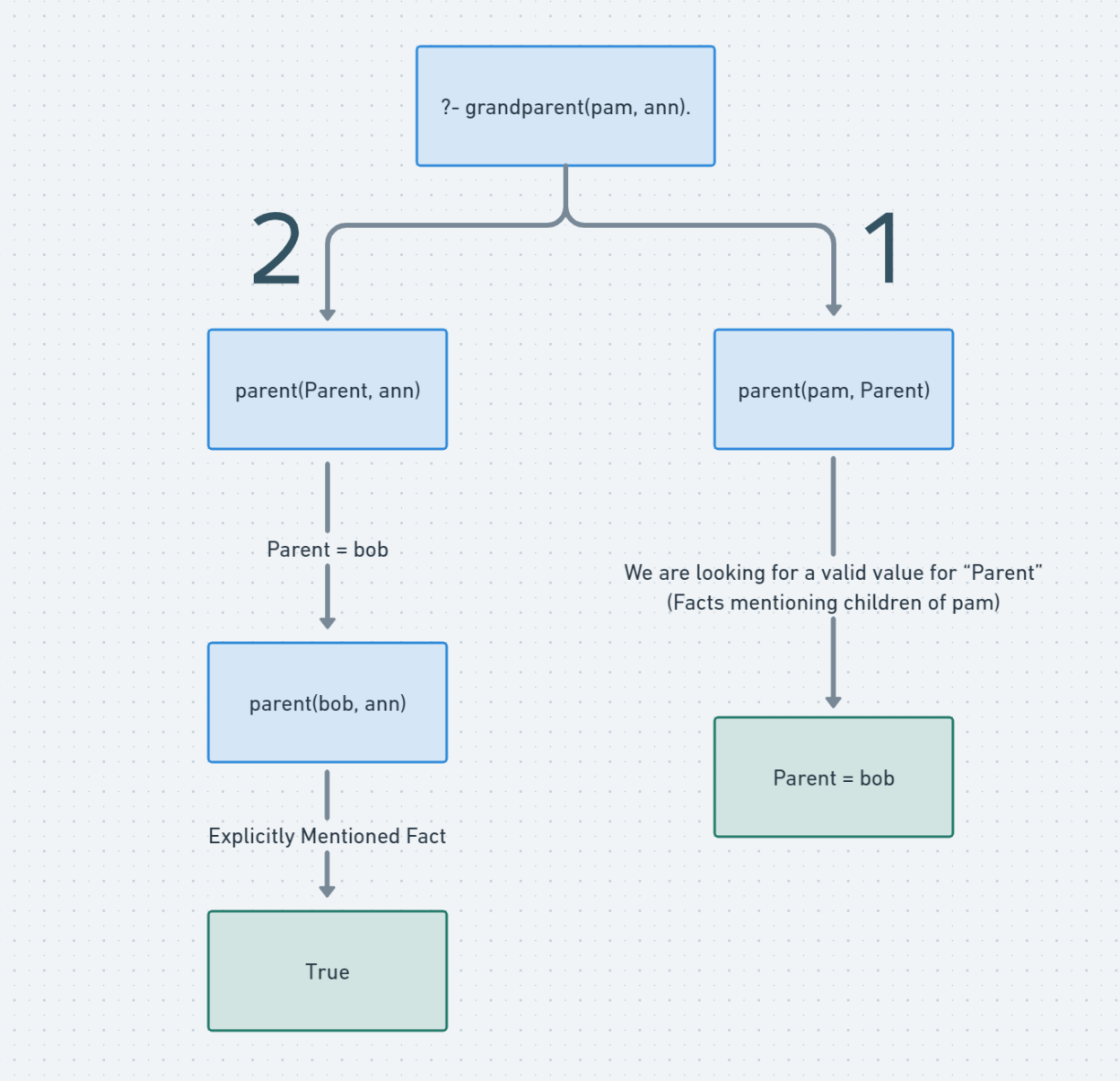
1. **?- parent(pam, bob).**



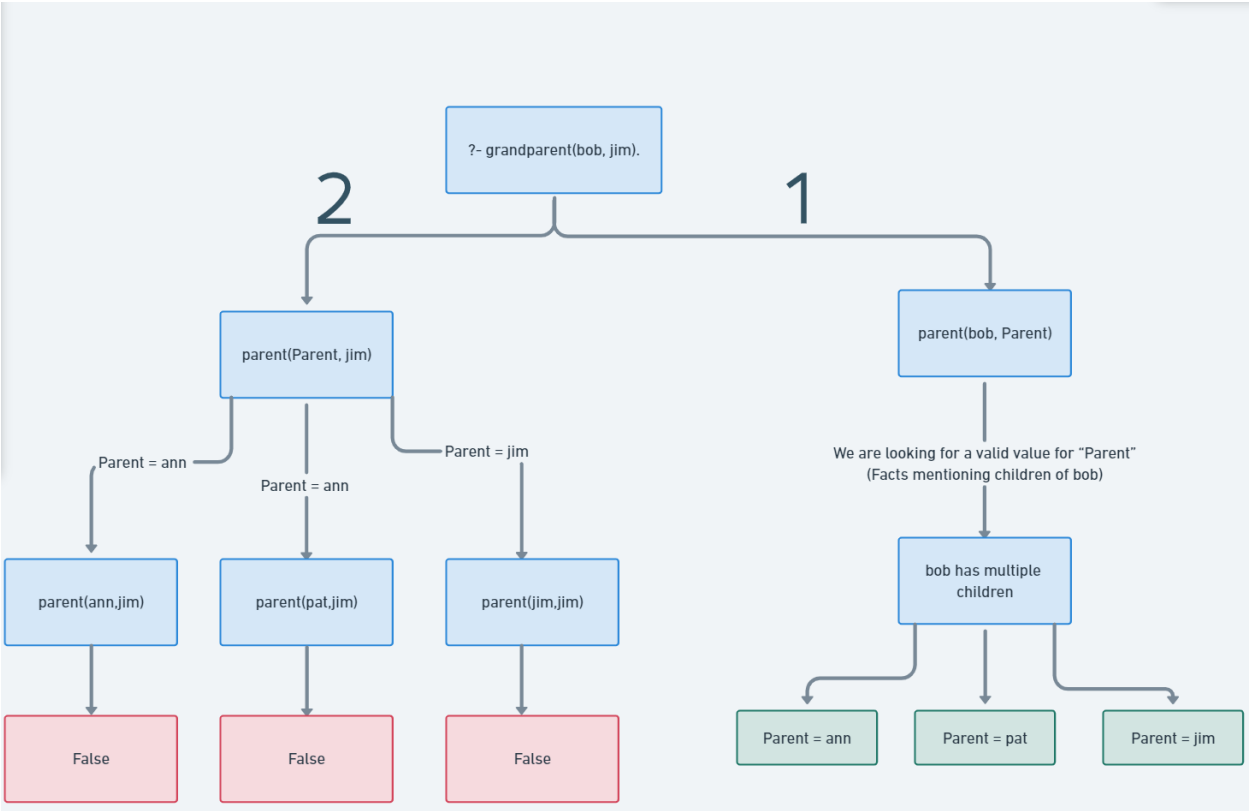
2. **?- mother(pam, bob).**



3. **grandparent(pam, ann).**



4. **?- grandparent(bob, jim).**



Specification:-

Processor	Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz 2.59 GHz
Installed RAM	16.0 GB (15.8 GB usable)

Appendix:-

```
parent(pam, bob).  % Pam is a parent of Bob
parent(tom, bob).
parent(tom, liz).
parent(bob, ann).
parent(bob, pat).
parent(bob, jim).
female(pam).              % pam is a female
male(tom).                % Tom is a male
male(bob).
female(liz).
female(ann).
female(pat).
male(jim).

mother(X, Y) :-           % X is the mother of Y if
        parent(X, Y),     % X is a parent of Y and
        female(X).   % X is female

predecessor(X, Y) :-      % Rule pr1: X is a predecessor of Y if
        parent(X, Y).     % X is a parent of Y
```

```
predecessor(X,Y) :-        % Rule pr2: X is a predecessor of Y if
        parent(X, Z),      % X is a parent of Z and
        predecessor(Z, Y). % Z is a predecessor of Y

grandchild(Grandchild, Grandparent) :- % Question 1
        parent(Grandparent, Child),
        parent(Child, Grandchild).

offspring(Y, X) :- % Question 4
        parent(X, Y).

grandparent(Grandparent, Grandchild) :- % Question 5
        parent(Grandparent, Parent),
        parent(Parent, Grandchild).

sister(Sister, Person) :- % Question 6
        female(Sister),
        parent(Parent, Sister),
        parent(Parent, Person),
        Sister \= Person.
```

```
happy(Person) :- % Question 7
        parent(Person, _).

has_two_children(Person) :- % Question 7
        parent(Person, Child1),
        parent(Person, Child2),
        sister(Sister, Child1),
        Child1 \= Child2.

grandchild(Grandchild, Grandparent) :- % Question 8
        parent(Parent, Grandchild),
        parent(Grandparent, Parent).

aunt(X, Y) :- % Question 9
    parent(Parent, Y),
    sister(X, Parent).
```