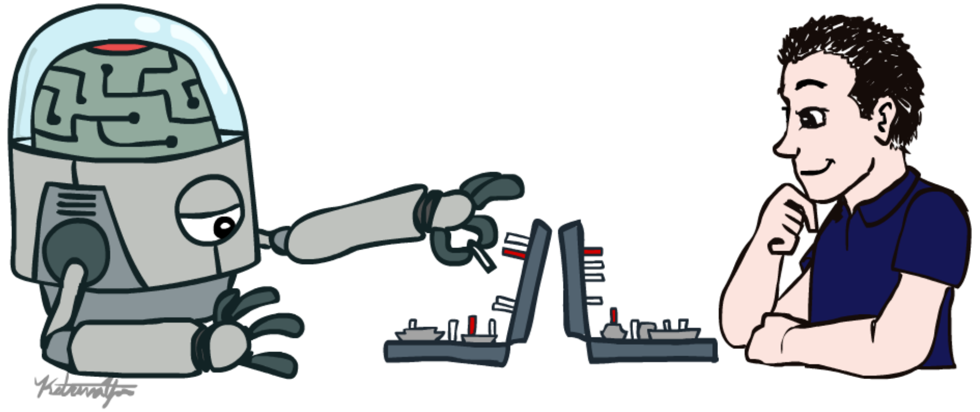
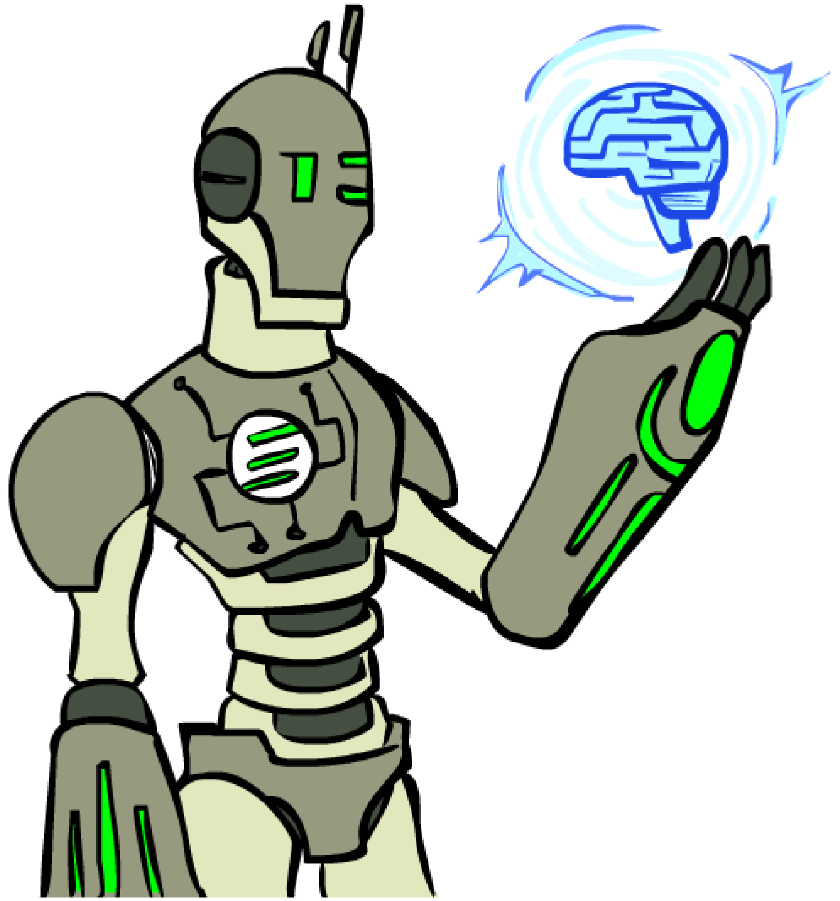
Lecture 08

Ashis Kumar Chanda
chanda@rowan.edu



Rowan University

Today



- Prolog with examples

Introduction

- Prolog is a logic programing language.
  - LOGical PROgramming → ProLog
- Used mainly for rapid prototyping language and symbol manipulation
- i.e., writing compilers, parsing natural language, automata theory.

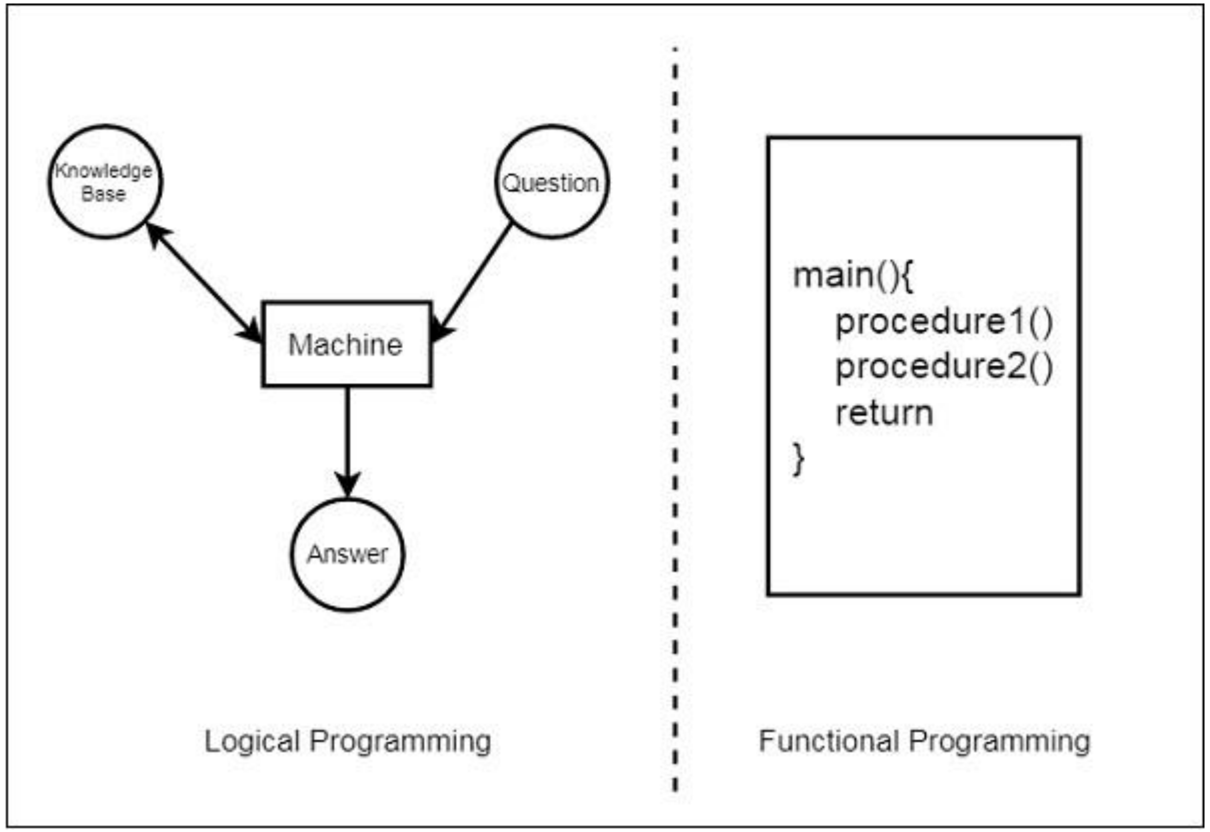
Introduction

- High level programming languages (i.e. C, Java, Python) use data structure for a domain specific procedure.
- It lacks a general mechanism for deriving facts from other facts.
- They also lack the expressiveness required to directly handle partial information (i.e. lack of any easy way to say some information).

Introduction

- Propositional logic is a declarative language.
- It has power to deal with partial information.
- **Compositionality**: The meaning of a sentence is a function of the meaning of its parts.

Logic and Functional Programming



```
main(){
   procedure1()
   procedure2()
   return
}
```

Basic parts

Prolog language basically has **three** different elements –

- **Facts** – The fact is predicate that is true,

For example, if we say, “Tom is the son of Jack”, then this is a fact.

- **Rules** – Rules are extinctions of facts that contain conditional clauses.
- **Questions** – And to run a prolog program, we need some questions, and those questions can be answered by the given facts and rules.

Facts

- The syntax for facts is as follows –
  - relation(object1,object2...).
  - **Example**:
  - cat(tom).
  - loves_to_eat(kunal,pasta).
  - of_color(hair,black).
  - loves_to_play_games(nawaz).
  - lazy(pratyusha).

Rules

- We can define rule as an implicit relationship between objects.
  - **rule_name**(object1, object2, ...) :- **fact/rule**(object1, object2, ...)
- Suppose a clause is like :
  - P :- Q;R.
- This can also be written as
  - P :- Q.
  - P :- R.

Rules: Example

- We can define rule as an implicit relationship between objects.
  - Lili is happy if she dances.
  - Tom is hungry if he is searching for food.
  - Jack and Bili are friends if both of them love to play cricket.

  - happy(lili) :- dances(lili).
  - hungry(tom) :- search_for_food(tom).
  - friends(jack, bili) :- lovesCricket(jack), lovesCricket(bili).

Queries

- Queries are some questions on the relationships between objects and object properties.
- So question can be anything, as given below –
  - Is tom a cat?
  - Does Kunal love to eat pasta?
  - Is Lili happy?
  - Will Ryan go to play?

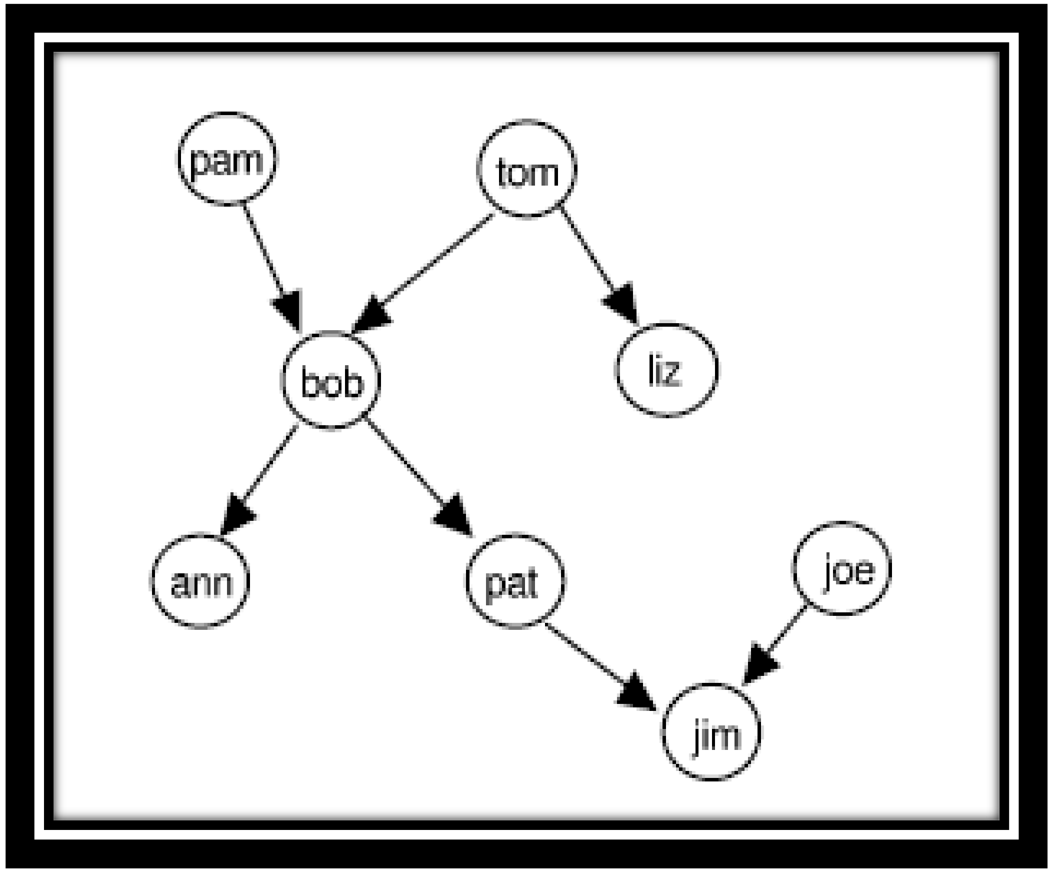
Prolog: syntax

- Uses uppercase letters for variables, lowercase for constants.
- Commas separate conjuncts.
- Implies are written as C :- A, B format,
  - Rather than A ^ B → C
- Don't use extra space
- Use % symbol for writing comments

https://swish.swi-prolog.org/

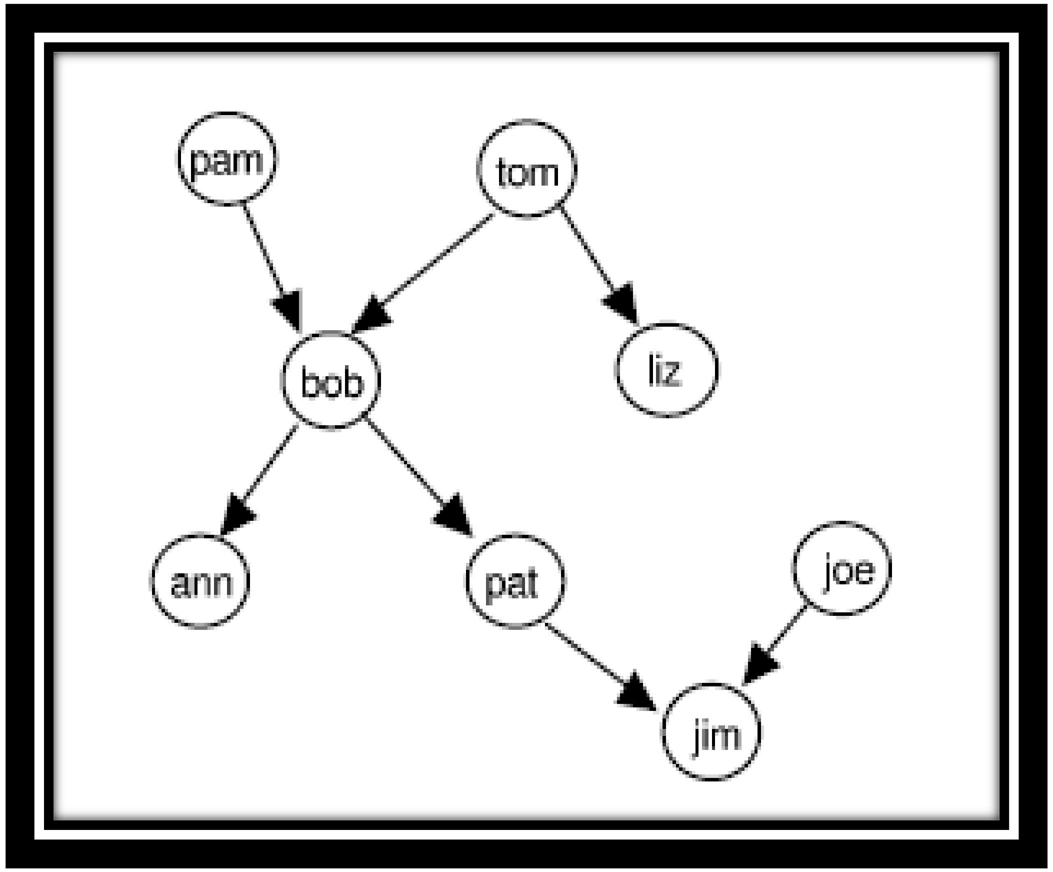
Example: Family Relationship in Prolog

- We want to make a family tree, and that will be mapped into facts and rules, then we can run some queries on them.



Example: Family Relationship in Prolog

parent (pam,bob)
parent (tom,bob)
parent (tom,liz)
parent (bob,ann)
parent (bob, pat)
parent (joe,jim)
parent (pat,jim)
female (pam)
male (tom)
male (bob)
female (liz)
female (ann)
female (pat)
male (joe)
male (jim)

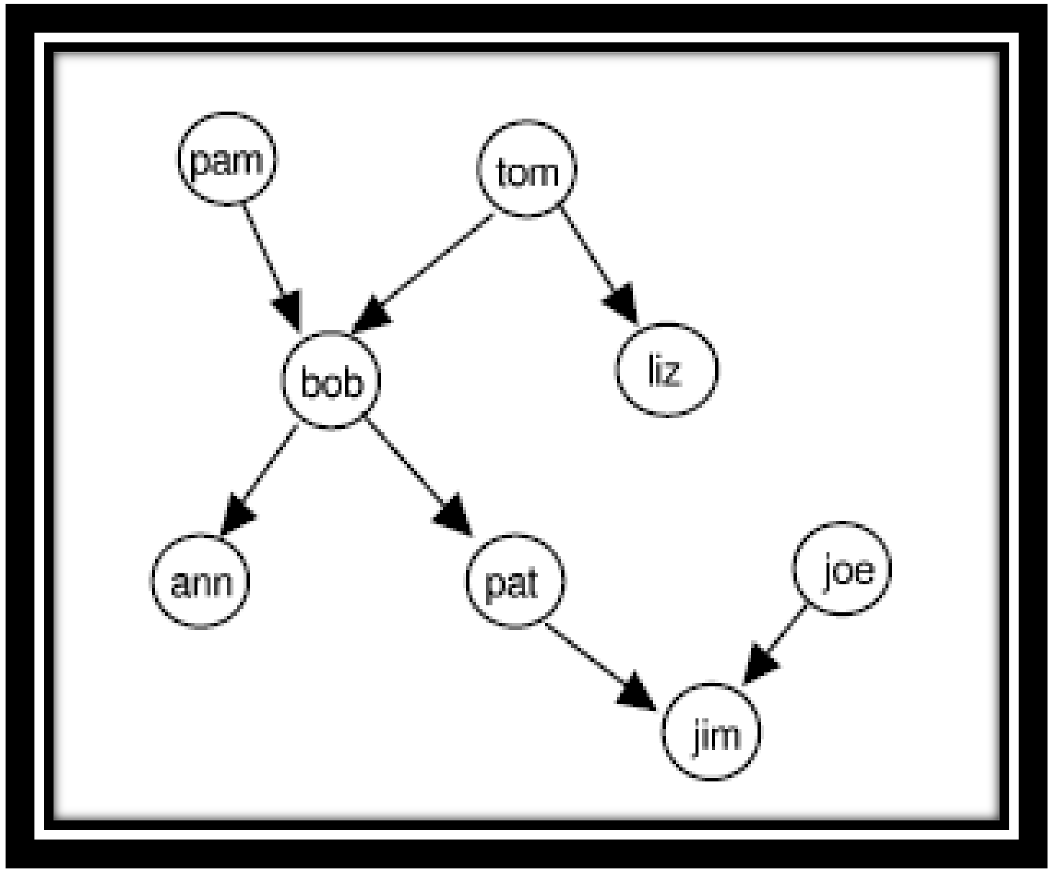


Example

```
|?- male (tom).
Yes

|?- male (X).
X=tom;
X= bob;
X=jim;

child (X,Y) :-  parent (Y,X)
|?- child (bob, A)
pam
tom
```



Example

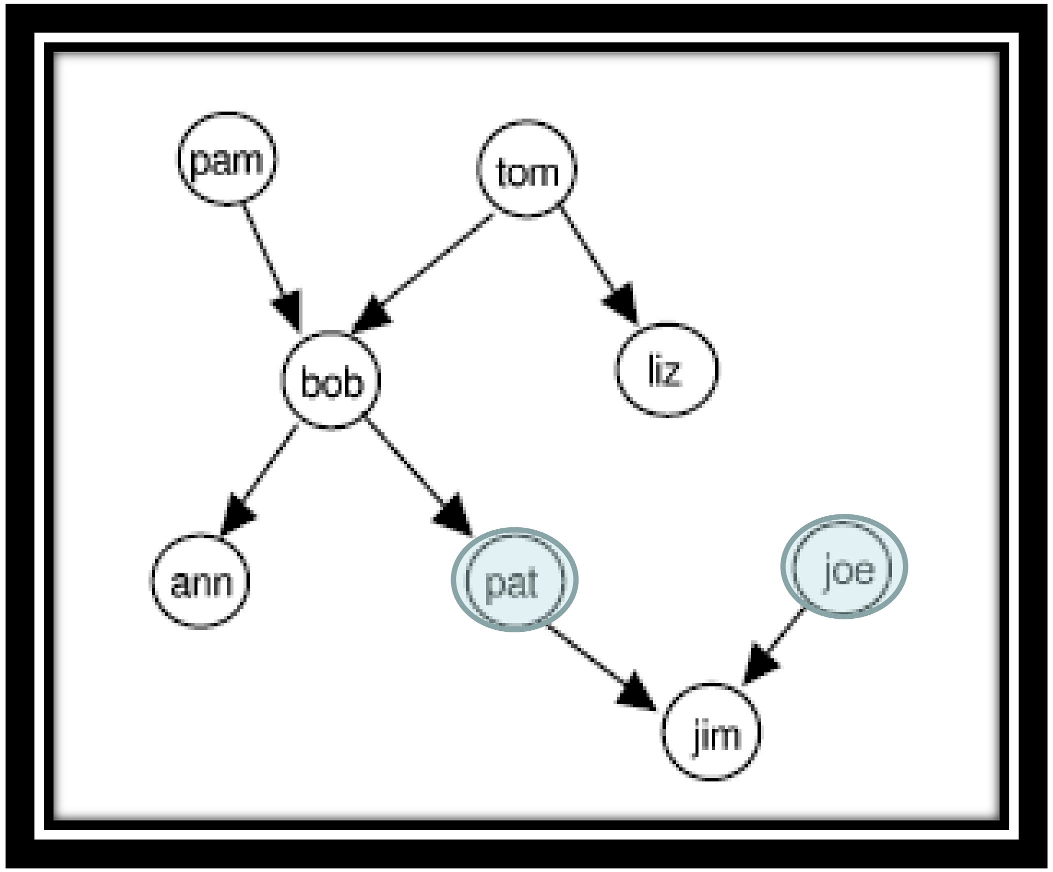
```
mother (X,Y) :- parent (X,Y) female (X).
father (X,Y ) :- parent (X,Y) male (X).

|?- mother (X, jim)
```

How prolog resolve ?

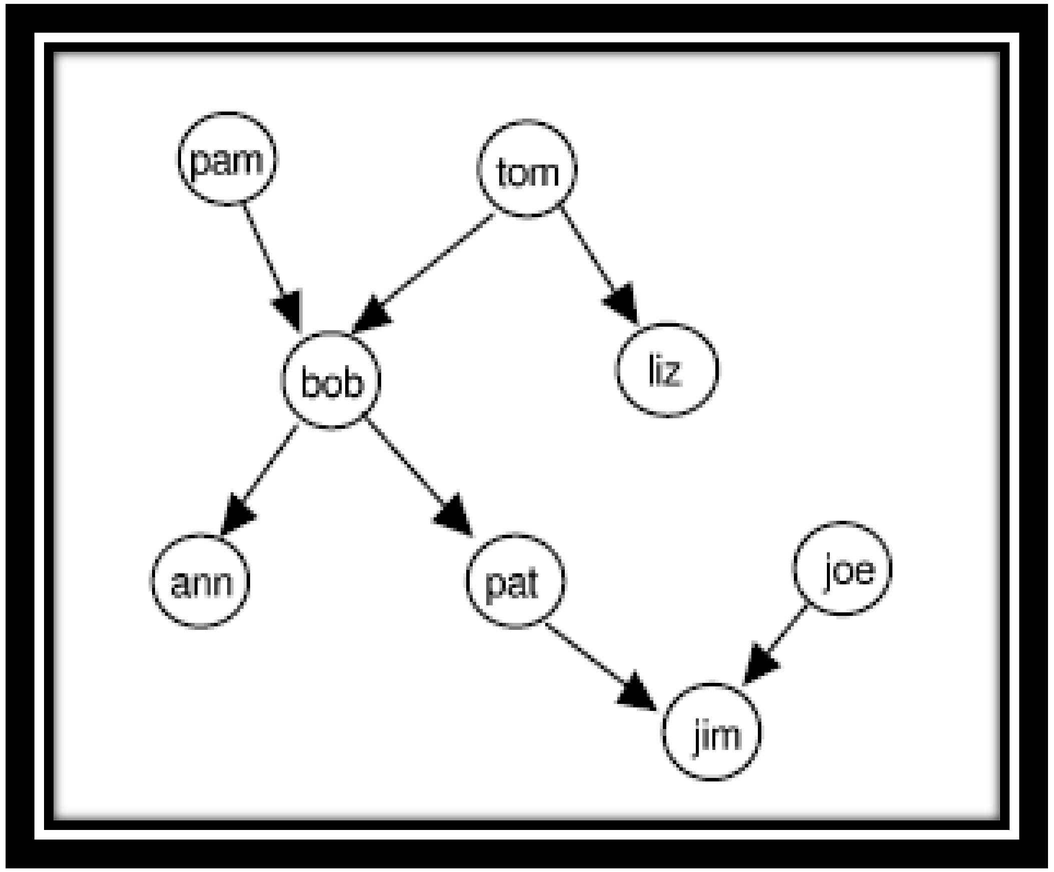
```
Mother (X, jim) :- parent (X, jim) female (X)
                :- parent (X, jim) female (X)
                :- female (pat)
                :-
```

X = pat



Example

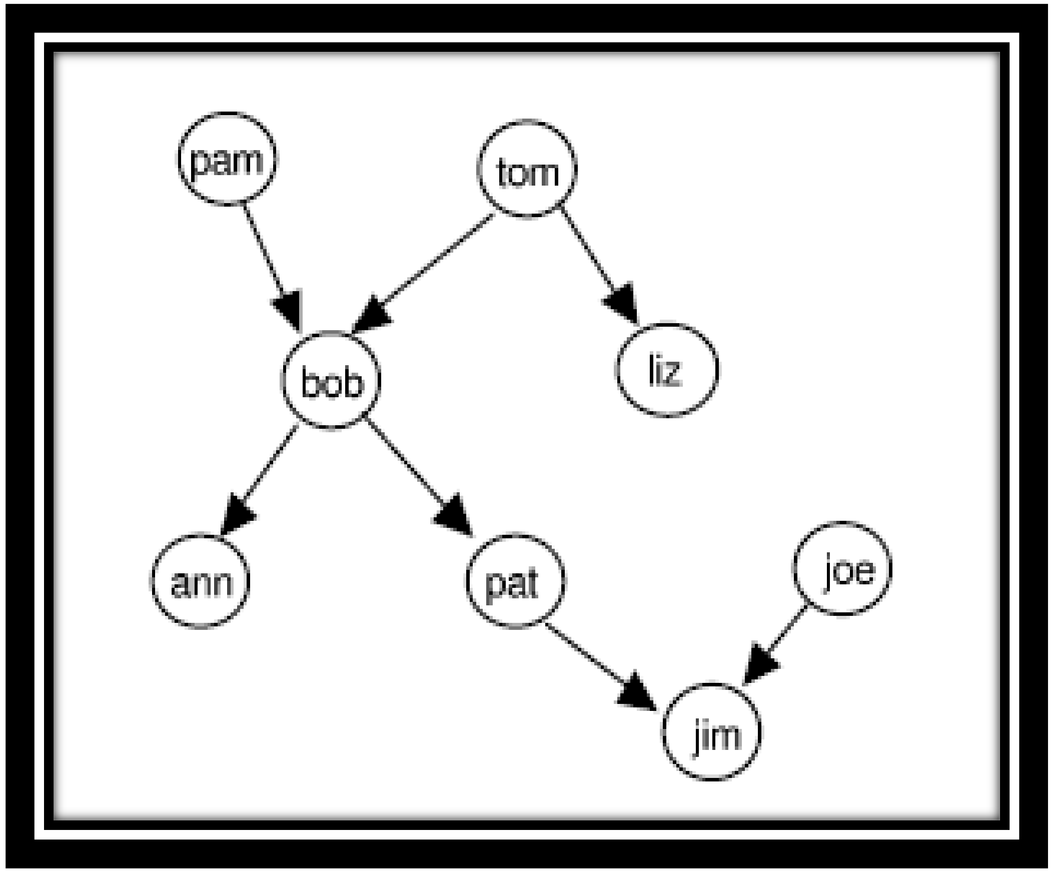
Write rules for brother.



brother(X,Y):-parent(Z,X),parent(Z,Y),male(X),**X\==Y.**

Example

Write rules for wife and grandfather.



```
wife(X,Y):-parent(X,Z),parent(Y,Z),female(X),male(Y).

grandfather(X,Z) :- father(X,Y), parent(Y,Z).
```

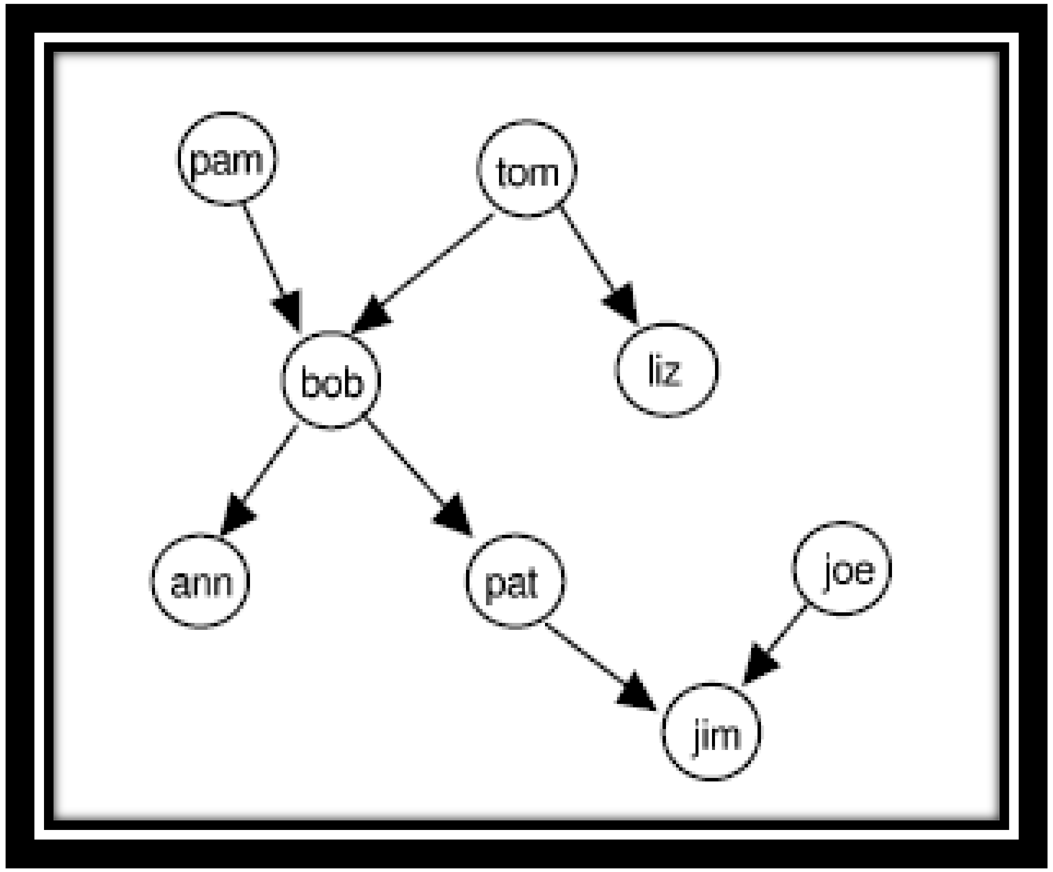
Example

grandparent(X,Z) :- parent(X,Y), parent(Y,Z).

|?- grandfather (X, pat).

ancestor (X,pat) :- parent (X,pat)

X=pam
X=tom



Example

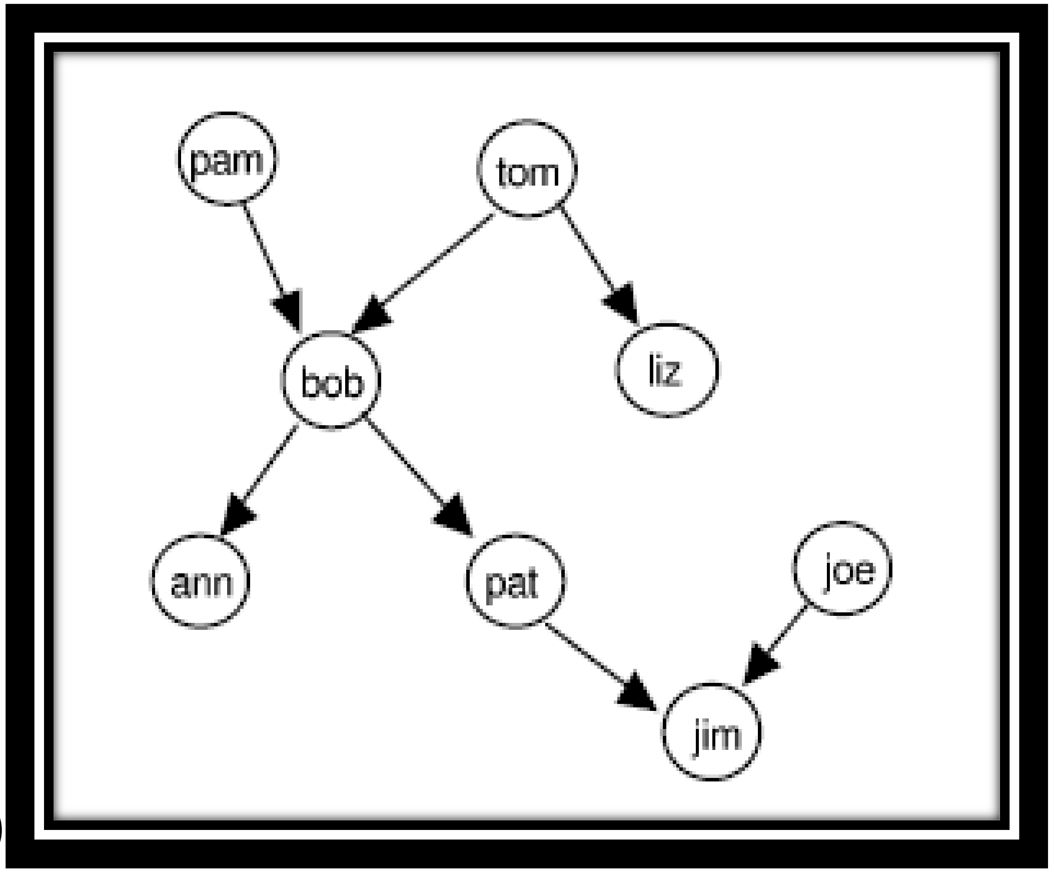
```
ancestor (X,Z) :- parent (X,Z).
ancestor (X,Z) :- parent (X,Y), ancestor (Y,Z).
```

recursive

```
|?- ancestor (X, pat).

ancestor (X,pat) :- parent (X,pat)

X=bob
ancestor (X,pat) :- parent (X,Y), ancestor (Y,pat)
                 :- parent (X,Y), ancestor (bob, pat)
Y=bob
X=pam
X=tom
```

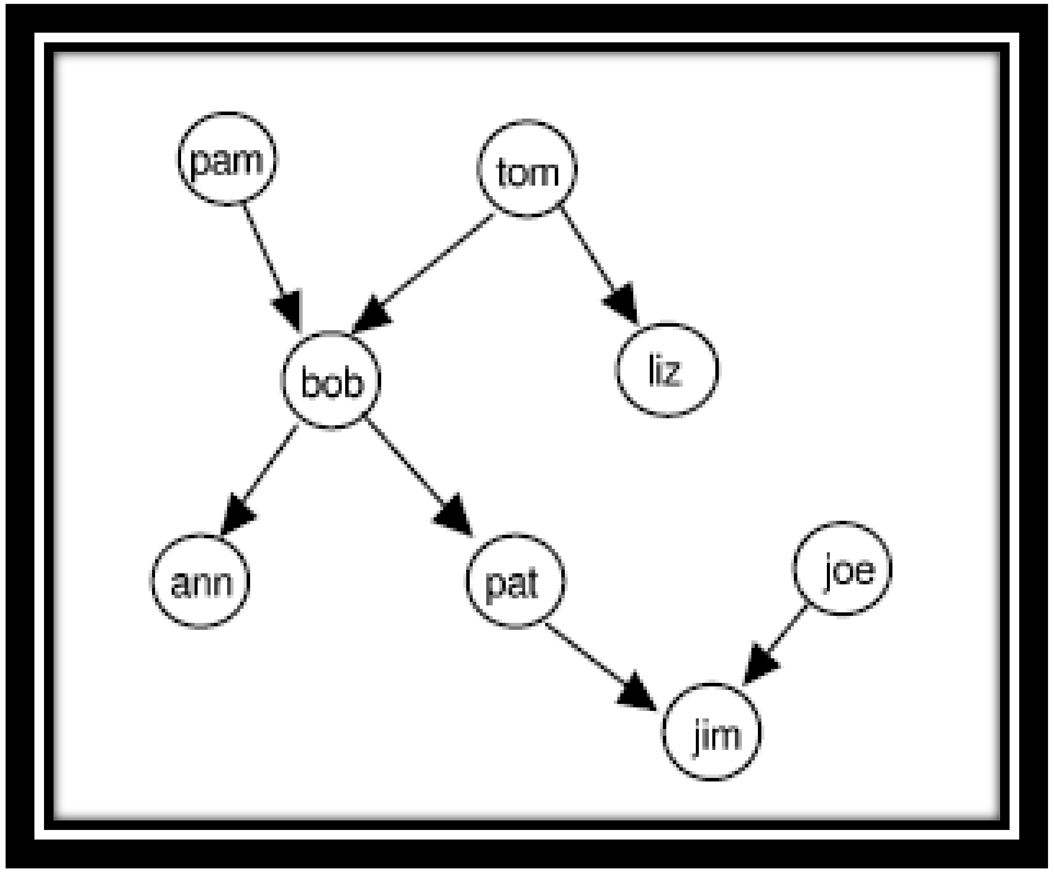


Example

```
ancestor (X,Z) :- parent (X,Z).
ancestor (X,Z) :- parent (X,Y), ancestor (Y,Z).


|?- ancestor (X, jim).
```

```
X = joe
X = pat
X = pam
X = tom
X = bob
```



Prolog is not logical !!

1. Ancestor (X,Z) :- ancestor (Y,Z), parent (X,Y).
2. Ancestor (X,Z) :- parent (X,Z).

?|- ancestor (X, pat)

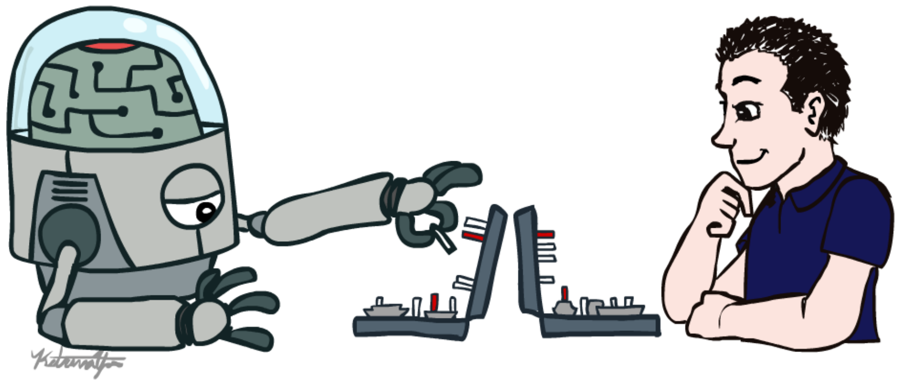
! Out of local stack during execution

[execution aborted]

Because : recursive call to the ancestor and Prolog always chooses the left hand literal first and always chooses the first program clause in the database.

More about Prolog

- You can also use different operators in prolog.
  - Operators: <, >, =, ! (not)
- Input and output options are also available in prolog.
- Useful for many real-life applications.
- Ex: Finding validity of a mathematical expression.
- More: “Prolog programming for AI” Textbook.



Thanks!