Introduction to Artificial Intelligence

Unit # 9

Induction vs. Deduction

- Deduction (General to Specific)
- Induction (Specific to General)
- Deduction
  - Given: All men are mortal (rule)
  - Shakespeare is a man (fact)
  - To Prove: Shakespeare is mortal (inference)
- Induction
  - Given: Shakespeare is mortal
  - Newton is mortal
  - Einstein is mortal (Observation)
  - To Prove: All men are mortal (Generalization)

Abduction vs. Deduction

- If there is rain, then there will be no picnic
- Example 1
  - Fact1: There was rain
  - Conclude: There was no picnic
- Example 2
  - Fact1: There was no picnic
  - Conclude: There was no rain (?)
- Induction and abduction are fallible forms of reasoning. Their conclusions are susceptible to retraction
- Two systems of logic
  - Propositional Calculus
  - Predicate Calculus

Why Reasoning?

- We judge intelligence of human beings by their ability to reason.
- Hard core research area of strong AI.
- Basis of many rule-based expert systems.
- Intelligent machines in Hollywood movies typically rely on their reasoning ability.

Propositional Logic

- Propositional Logic, also known as sentential logic, is a formal system in which knowledge is represented as propositions.
- A *proposition is a statement,* or a simple declarative sentence.
- For example, “Fish is expensive” is a proposition.
- In terms of binary logic, this proposition could be false in Karachi, but true in Lahore. But a proposition always has a truth value.

Deductive Reasoning

- In deductive reasoning, the conclusion is reached from a previously known set of premises.
- If the premises are true, then the conclusion must also be true.
  - If it’s raining, the ground is wet.
  - If the ground is wet, the ground is slippery.
- These are also inference rules that will be used in deduction.
- Now we introduce another premise that
  - It is raining.
- Now, let’s prove that it’s slippery.

Predicate (First-Order) Logic

- Propositional logic is useful but it cannot represent general-purpose logic in a compact and succinct way.
- Using FOL, we can use both predicates and variables to add greater expressiveness as well as more generalization to our knowledge.
- In FOL, knowledge is built up from constants (the objects of the knowledge), a set of predicates (relationships between the knowledge), and some number of functions (indirect references to other knowledge).

First-order logic

- Whereas propositional logic assumes the world contains facts,
- first-order logic (like natural language) assumes the world contains
  - Objects: people, houses, numbers, colors, baseball games, wars, …
  - Relations: red, round, prime, brother of, bigger than, part of, comes between, …
  - Functions: father of, best friend, one more than, plus, …

Using FOL

- Brothers are siblings

  $\forall x,y\ Brother(x,y) \Leftrightarrow Sibling(x,y)$
- One's mother is one's female parent

  $\forall m,c\ Mother(m) \Leftrightarrow (Female(m) \wedge Parent(m,c))$
- "Sibling" is symmetric

  $\forall x,y\ Sibling(x,y) \Leftrightarrow Sibling(y,x)$
- Every gardener likes the sun.

  $\forall x$ gardener(x) => likes (x, Sun)

Prolog

- Prolog is a logic programming language.
- Programming languages are of two kinds:
  - **Procedural** (BASIC, ForTran, C++, Pascal, Java);
  - **Declarative** (LISP, Prolog, ML, SQL).
- In procedural programming, we tell the computer **how** to solve a problem.
- In declarative programming, we tell the computer **what** problem we want solved.

Programming in Prolog

- Computer programming in Prolog consists of:
  - Specifying some facts about objects and their relationships,
  - Defining some rules about objects and their relationships, and
  - Asking questions about objects and their relationships

Basic Elements of Prolog

- Our program is a database of **facts** and **rules**.
- Some are always true (facts):

  father(john, jim).

- Some are dependent on others being true (rules):

  parent(Person1, Person2) :-
  father(Person1, Person2).

- To run a program, we ask questions about the database.

Predicate Definitions

- Both facts and rules are predicate definitions.
- *'Predicate'* is the name given to the word occurring before the bracket in a fact or rule:

```
parent(jane,alan).
```

Predicate name

- By defining a predicate you are specifying which information needs to be known for the property denoted by the predicate to be true.

Clauses

- Predicate definitions consist of *clauses.*
  = An individual definition (whether it be a fact or rule).

e.g.
```
mother(jane,alan).
```
= Fact

```
parent(P1,P2):- mother(P1,P2).
```
= Rule

head **body**

- A clause consists of a *head*
- and sometimes a *body.*
  – Facts don't have a body because they are always true.

Arguments

- A predicate head consists of a *predicate name* and sometimes some *arguments* contained within brackets and separated by commas.

```
mother(jane,alan).
```

Predicate name **Arguments**

- A body can be made up of any number of subgoals (calls to other predicates) and terms.
- Arguments also consist of terms, which can be:
  - Constants e.g. jane,
  - Variables e.g. Person1, or

Prolog in English

Example Database:

John is the father of Jim.
Jane is the mother of Jim.
Jack is the father of John.

Person 1 is a parent of Person 2 **if**
Person 1 is the father of Person 2 **or**
Person 1 is the mother of Person 2.

Person 1 is a grandparent of Person 2 **if**
some Person 3 is a parent of Person 2 **and**
Person 1 is a parent of Person 3.

Facts

Rules

Example questions:

Who is Jim's father?
Is Jane the mother of Fred?
Is Jane the mother of Jim?
Does Jack have a grandchild?

Prolog in Prolog

Example Database:

John is the father of Jim.
Jane is the mother of Jim.
Jack is the father of John.

Person 1 is a parent of Person 2 **if**
Person 1 is the father of Person 2 **or**
Person 1 is the mother of Person 2.

Person 1 is a grandparent of Person 2 **if**
some Person 3 is a parent of Person 2 **and**
Person 1 is a parent of Person 3.

Example questions:

Who is Jim's father?
Is Jane the mother of Fred?
Is Jane the mother of Jim?
Does Jack have a grandchild?

Example Database:

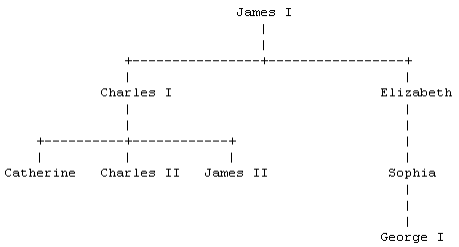
```
father( john, jim ).
mother( jane, jim ).
father( jack, john ).

parent( Person1, Person2 ) :-
        father( Person1, Person2 ).
parent( Person1, Person2 ) :-
        mother( Person1, Person2 ).

grandparent( Person1, Person2 ) :-
        parent( Person3, Person2 ),
        parent( Person1, Person3 ).
```

Example questions:

```
?- father( Who, jim ).
?- mother( jane, fred ).
?- mother( jane, jim ).
?- grandparent( jack, _ ).
```


Practice Question

```
male(james1). male(charles1).
male(charles2). male(james2).
male(george1). female(catherine).
female(elizabeth). female(sophia).
parent(james1, charles1).
parent(james1, elizabeth).
parent(charles1, charles2).
parent(charles1,catherine).
parent(charles1, james2).
parent(elizabeth, sophia).
parent(sophia, george1).
```



Write Prolog statements to answer the following queries

Was George I the parent of Charles I?
Who was Charles I's parent?
Who were the children of Charles I?
M is the mother of X if she is a parent of X and is female
F is the father of X if he is a parent of X and is male
X is a sibling of Y if they both have the same parent.

Prolog Queries

- Was George I the parent of Charles I?
  - parent(geroge1, charles1).
- Who was Charles I's parent?
  - parent(X, charles1).
- Who were the children of Charles I?
  - parent(charles1, X).
- M is the mother of X if she is a parent of X and is female
  - mother(M,X) :- female(M), parent(M,X).
- F is the father of X if he is a parent of X and is male
  - father(F,X) :- female(F), parent(F,X).
- X is a sibling of Y if they both have the same parent.
  - sibling(X,Y) :- parent(Z,X), parent(Z,Y), X \= Y.
  - X \= Y is equivalent to "X not equals to Y". Try to run the program without adding this statement and then add this statement and see the difference in the output.

Example

- Suppose someone has already written Prolog clauses that define the following relationships:
  - father(X, Y) /* X is the father of Y */
  - mother(X, Y)
  - male(X)
  - female(X)
  - parent(X, Y)
- Write Prolog clauses to define the following relationships:
  - Is_mother(X), is_father(X), is_son(X), sister_of(X, Y), grandpa_of(X, Y), sibling(X, Y)
- Example
  - Aunt(X, Y) :- sister_of(X, Z), parent(Z, Y)

Crime Scene

```
% The possible suspects of the crimes are:
possible_suspect(fred).
possible_suspect(mary).
possible_suspect(jane).
possible_suspect(george).

% The facts about the crimes from the police log.
crime(robbery1, john, tuesday, park).
crime(assault1, mary, wednesday, park).
crime(robbery2, jim, wednesday, pub).
crime(assault2, robin, thursday, park).
```


Crime Scene (Cont'd)

```
% Tell prolog where the suspects were on different days.
was_at(fred, park, tuesday).
was_at(fred, pub, wednesday).
was_at(fred, pub, thursday).

was_at(george, pub, tuesday).
was_at(george, pub, wednesday).
was_at(george, home, thursday).

was_at(jane, home, tuesday).
was_at(jane, park, wednesday).
was_at(jane, park, thursday).

was_at(mary, pub, tuesday).
was_at(mary, park, wednesday).
was_at(mary, home, thursday).
```

Crime Scene (Cont'd)

```
% Tell prolog who is jealous of who
jealous_of(fred, john).
jealous_of(jane, mary).

% And who owes money to whom
owes_money_to(george, jim).
owes_money_to(mary, robin).

% A Person has a motive against a Victim if Person is jealous of
% Victim or Person owes money to Victim.

% A Person is a prime suspect of a crime if Person is a possible suspect and
% the person was at the time and place of the crime and the person had a
% motive against the victim of the crime.
```


Crime Scene (Cont'd)

```
% A Person has a motive against a Victim if Person is jealous of
% Victim or Person owes money to Victim.
motive_against(Person, Victim) :-
    jealous_of(Person, Victim);
    owes_money_to(Person, Victim).

% A Person is a prime suspect of a crime if Person is a possible suspect and
% the person was at the time and place of the crime and the person had a
% motive against the victim of the crime.
prime_suspect(Person, Crime) :-
  possible_suspect(Person),
  was_at(Person, Place, Day),
  crime(Crime, Name, Day, Place),
  motive_against(Person, Name).
```

Solar Systems

```
% facts about planets
orbits(mercury, sun).
orbits(venus, sun).
orbits(earth, sun).
orbits(mars, sun).
% facts about moons
orbits(moon, earth).
orbits(phobos, mars).
orbits(deimos, mars).

% An object P is a planet if it orbits sun.
% An object S is a satellite if it orbits a planet P
```


Course Prerequisites

```
prereq(introCS,dataStructs).
prereq(dataStructs,progLangs).
prereq(dataStructs,graphics).
prereq(linAlg,graphics).

%  A course R is a required course for course C if R is a prerequisite of
%  C.

% A course R is a required course for course C if R is a prerequisite of
% some course S which is a required course for R.
```