UNIT-IV Part 3 (of 3)
Case Study on Logic Programming with Prolog

TOPICS: Intro to logic
Intro to Logic Programming and Prolog
BASIC ELEMENTS of PROLOG
Arithmetic Example
Prolog Deficiencies

Recap: Intro to logic

Basic Elements of First Order Logic

Constant	1, 2, A, John, Mumbai, cat,....
Variables	x, y, z, a, b,....
Predicates	Brother, Father, >,....
Function	sqrt, LeftLegOf, ...
Connectives	$\wedge$, $\vee$, $\neg$, $\Rightarrow$, $\Leftrightarrow$
Equality	==
Quantifier	$\forall$, $\exists$

Atomic sentences:

- Atomic sentences are the most basic sentences of first-order logic. These sentences are formed from a predicate symbol followed by a parenthesis with a sequence of terms.
- We can represent atomic sentences as **Predicate (term1, term2, ..., term n)**.
- **Example: Ravi and Ajay are brothers: => Brothers(Ravi, Ajay).**

  Chinky is a cat: => cat (Chinky)

Complex Sentences:

- Complex sentences are made by combining atomic sentences using connectives.
- **First-order logic statements can be divided into two parts:**
- **Subject:** Subject is the main part of the statement.
- **Predicate:** A predicate can be defined as a relation, which binds two atoms together in a statement.
- **Consider the statement: "x is an integer."**, it consists of two parts, the first part x is the subject of the statement and second part "is an integer," is known as a predicate.

Quantifiers in First-order logic:

- A quantifier is a language element which generates quantification, and quantification specifies the quantity of specimen in the universe of discourse.
- These are the symbols that permit to determine or identify the range and scope of the variable in the logical expression. There are two types of quantifier:
  - **Universal Quantifier, (for all, everyone, everything)**
  - **Existential quantifier, (for some, at least one).**

Universal Quantifier:

- Universal quantifier is a symbol of logical representation, which specifies that the statement within its range is true for everything or every instance of a particular thing.
- The Universal quantifier is represented by a symbol ∀, which resembles an inverted A
- Note : In universal quantifier we use implication "→".
- **∀x man(x) → drink (x, coffee).** It will be read as: There are all x where x is a man who drink coffee

Existential Quantifier:

- Existential quantifiers are the type of quantifiers, which express that the statement within its scope is true for at least one instance of something.
- It is denoted by the logical operator ∃, which resembles as inverted E. When it is used with a predicate variable then it is called as an existential quantifier.

Existential Quantifier:

- If x is a variable, then existential quantifier will be $\exists x$ or $\exists(x)$. And it will be read as:
- **There exists a 'x.'**
- **For some 'x.'**
- **For at least one 'x.'**

Existential Quantifier

- In Existential quantifier we always use AND or Conjunction symbol ($\wedge$)

- **$\exists$x: boys(x) $\wedge$ intelligent(x)**
- It will be read as: There are some x where x is a boy who is intelligent.

Points to remember:

- The main connective for universal quantifier $\forall$ is implication $\rightarrow$.
- The main connective for existential quantifier $\exists$ is and $\wedge$.

Properties of Quantifiers:

- In universal quantifier, $\forall x \forall y$ is similar to $\forall y \forall x$.
- In Existential quantifier, $\exists x \exists y$ is similar to $\exists y \exists x$.
- $\exists x \forall y$ is not similar to $\forall y \exists x$.

Intro to Logic Programming and Prolog

Motivation: Logic Programming

1. Reduce the programming burden.

2. System should simply accept the necessary information and the objective (goal), and then figure out its own solution.

3. Have a program that looks more like its own specification.

4. Take advantage of logical inference to automatically get many of the consequences of the given information.

Prolog

- Important role in Artificial Intelligence
- Intended primarily as **a declarative programming language**
- It is a logic-based language, its syntax is based on Horn clauses, a predicate form derived from the predicate calculus
- Logic is expressed as **relations (called as Facts and Rules)**
- Core heart = Logic being applied

“Prolog” --- “**Pro**gramming in **lo**gic”

…Prolog

- Prolog is an example of programming language **designed to reason** using rules of predicate logic
- **Prolog programs** include a set of declarations consisting of two types of statements - Prolog facts and Prolog rules

Prolog : a declarative language

- Prolog programs need only to **describe** goals and **provide** facts and rules for the Prolog system to use in searching for a goal solution

- Compare to imperative languages, which describe algorithms and specify the exact steps required to achieve a goal

Basic Elements of Prolog

Sebesta Pg 614

Topics: Terms, Fact Statements, Rule Statement, Inferencing Process of Prolog

Prolog Statements

- Prolog programs are a collection of statements
- There a only few kinds of statements in Prolog (But can be complex)
- All Prolog statements are constructed from **TERMS**

TERMS

Prolog TERM is a CONSTANT, VARIABLE, or STRUCTURE

Facts, rules, and queries built out of TERMS

4 TERMS in PROLOG

TERMS= CONSTANT(atom,interger)/ VARIABLE / STRUCTURE

CONSTANT = **ATOM** , **INTEGER(NUMBERS)** **(2)**

VARIABLES = Start with UPPERCASE Letter, _ **(1)**

COMPLEX TERMS (or STRUCTURES) =has form**(2)**

ATOM (*not dividable into smaller parts*)

- Symbolic values of Prolog
- Consists of : String of letters, digits or underscores beginning with lowercase letter

  OR

  String of Any printable ASCII character delimited with ‘ ’

Examples of Atoms

Atom	Not and Atom
‘this is an atom’	This is not an atom
‘this%too’	This%tooisnot
abcd23	Abcd23
Parent	Parent
Likes	Likes
jOHN	John

Variables

- Consists of Letters, Digits and underscore and starting with Uppercase Letter
- For Variables starting with _ (underscore) Prolog systems make internal use of such variable names, e.g. _G157
- Sample variables -

  Var_123, _G17, X, Y, Who, Somebody….

Structures

- They are the Atomic Propositions of Predicate Calculus
- Their general form is **functor(parameter list)**
  - Functor is an Atom and is used to identify the structure
  - Parameter list is list of either Atoms, Variables, or other structure

…structures

- They are means of specifying FACTS in Prolog
- They can also be thought of as objects
  - They allow facts to be stated in terms of several related stoms
  - In this sense, structures are relations, as they state relationships among terms
- It is also a Predicate, when its context specifies it to be a query

Statements in Prolog

- 2 basic statement forms for writing program
  - Facts
  - Rules

- Query, to find new things

So, only **three basic constructs** in Prolog: Facts, Rules and Queries

Prolog Sentences

likes(john, sam). /*John likes Susie */

likes(X, cake). /* Everyone likes Susie */

likes(john, Y). /* John likes everybody */

Q) Convert to Prolog Sentences

/* John likes everybody and everybody likes John */

/* John likes Susie or John likes Mary */

/* John does not like pizza */

/* John likes Susie if John likes Mary.

Ans

- likes(john, Y), likes(Y, john).

- likes(john, susie); likes(john,mary).

- not(likes(john,pizza)).

- likes(john,susie) :- likes(john,mary).

Fact Statements

- Simple statements assumed to be true
- Eg
  - female(shelly).
  - female(mary).
  - male(bill).
  - male(jake).
  - father(bill,jake).

Note: there is period (.) after every fact

Basic construct in Prolog: Facts

FACTS

- mstudent(rahul).
- mstudent(rohan).
- mstudent(rohini).
- isFree(rohini).
- isFree(rohan).
- playsCricket(rahul).

- Facts are part of Knowledge base (KB)

Creating a Knowledge Base (KB1)

Swi-prolog -> kb1.pl

Knowledge Base (KB1)

```
mstudent(rahul).
mstudent(rohan).
mstudent(rohini).
isFree(rohini).
isFree(rohan).
playsCricket(rahul).
```

PREDICATES (or PROCEDURES)

3 Predicates ---mstudent, isFree, playsCricket

Rule Statements

- Form corresponds to headed Horn clause
- Form of Rule Statement

  LHS :- RHS

  {consequent}:- {antecedent}

  Consequent - is single term (as per Horn clause)

  Form in Prolog

  Consequent_1:-antecedent_expression

…contd… Rule Statements

Consequent_1:-antecedent_expression

- General form can be read as –

  "Consequent_1 can be concluded if the antecedent expression is true, or can be madetrue by some instantiation of its variable"

E.g.

ancestor(mary, shelly):-mother(mary,shelly)

….Rule Statement …

- Antecedent Expression
  - In Prolog, atomic propositions in a conjunction are separated by ‘,’ comma, so comma can be considered AND operation
  - E.g. female(shelly),child(shelly)

Sample Rule Statement

ancestor(mary,shelly):-mother(mary,shelly).

This states that, if mary is mother of shelly, then mary is ancestor of shelly.

(*Headed horn clause are called RULES because they state rules of implication between propositions*)

RULES

- Rules state information that is conditionally true of the domain of interest
- Rule

  head :- body

- Prolog knows that body follows from the information in the knowledge base, then Prolog can infer head.
- **This fundamental deduction step is what logicians call modus ponens.**

Modus Ponens

- The rule of logic which states that if a conditional statement (‘if *p* then *q*’) is accepted, and the antecedent (*p*) holds, then the consequent (*q*) may be inferred.

Knowledge Base (KB2)- Rules

```
mstudent(rahul).
mstudent(rohan).
mstudent(rohini).
isFree(rohan).
isFree(rohini).
playsCricket(rahul).
playsCricket(rohan):-isFree(rohan).
playsGuitar(rohini):-isFree(rohini).
isHappy(rohan):-playsCricket(rohan).
```

CLAUSES

- The facts and rules contained in a knowledge base are called clauses.
- Knowledge base (KB2) above has 9 clauses - 6 facts and 3 rules

More on PREDICATES

Predicates are Important concepts.

Clauses concerning them - define what they mean and how they are inter-related

Predicates Definition

- Predicate **mstudent** is defined using 3 clauses(facts)
- Predicate **isFree** is defined using 2 clauses (facts)
- Predicate **playsCricket** is defined using 2 clauses (1 fact, 1 rule)
- Predicate **playsGuitar** is defined using 1 clause(rule)
- Predicate **isHappy** is defined using 1 clause(rule)

Prolog logic:

- the :- means implication
- the , means conjunction
- and the ; means disjunction
- Standard logical proof rule (**modus ponens**) plays an important role

KNOWLEDGE BASE (KB3) with 5 CLAUSES- 2 FACTS and 3 RULES

KNOWLEDGE BASE (KB3) has 3 Predicates (isFree, likesCricket,..)

```
isFree(rakesh).
likesCricket(rishi).
playsCricket(rakesh):-
        likesCricket(rakesh),
        isFree(rakesh).
playsCricket(rishi):-
        likesCricket(rishi).
playsCricket(rishi):-
        isFree(rishi).
```

More on KB3

- Predicates of KB3 are isFree, likesCricket, playCricket
- NOTE: Rule playsCricket(rakesh) has COMMA and 2 items (GOALS) in its body
- NOTE: logical conjunction is expressed in Prolog as COMMA (comma means AND)
- Above rule is - Rakesh plays criket if he likes cricket AND he is Free.

More on KB3

Thus, query below give "no"

?- playsCricket(rakesh).

OR /either condition (logical Disjunction) is expressed by having 2 separate rules, like rule playsCricket(rishi)

Thus, query below give "yes"

?- playsCricket(rishi).

OR /either condition (logical Disjunction) can also be expressed bwith ";" a semicolon, e.g.

playsCricket(rishi):- likesCricket(rishi); isFree(rishi).

Use of Variables in Prolog Statements

Rules of implication among some variables or **universal objects** (*here X, Y, M, F*)

```
parent(X,Y) :- mother(X,Y)
parent(X,Y):- father(X,Y)
sibling(X,Y):- mother(M,X),mother(M,Y),
               father(F,X), father(F,Y)
```

QUERY / Goal Statement

- After writing Program, the **Goal/QUERIES** is required to be proved or Disproved.
- Form of **goal/query** is like headless horn clause
  - E.g. parent(fred)

QUERIES

```
mstudent(rahul).
mstudent(rohan).
mstudent(rohini).
isFree(rohini).
isFree(rohan).
playsCricket(rahul).
```

?- mstudent(rahul)

Prolog answer

yes

……Goal Statement

- System response is yes or no;
  - yes means goal/query is proved TRUE, under the given database of FACTS and RULES
  - no means that either Goal/Query is proved FALSE or system could not prove/disprove it
- Conjunctive Proposition are also Legal Goals
  - Each proposition in it is a **SUBGOAL**
- Proposition with Variables are also Legal Goals
  - E.g. father(X,mike)   which also gives X=val

    where val is the value to which X was temporarily instantiated through UNIFICATION

VARIABLES in QUERIES

Words starting with Upper-case letter is a Prolog variable

KB4

```
food(cake).
food(pasta).
food(burger).
likes(rohan,cake).
likes(rishi,cake).
likes(rama,burger).
likes(rohan,burger).

?- food(X).
X = cake
```

Inference Rules for PROLOG

- **No existential quantifiers used**
- **Only Universally Quantified Variables are used**
  - **Rules of Inference typically say :-**
    - **“if there is an x such that P(x) is True, then Q(x) holds too”**
    - **Equivalently : For all x such that P(x) is true, Q(x) is also True**
    - **Rules are expressed as Implication Statements**

      { antecedents } -> consequent

2 Primary Operations used by Inference Rules to Deduce new facts

- Resolution
- Unification

Inferencing Process of Prolog

IMPORTANT SUB POINTS:
Inferencing (Resolution) Process
2 Approaches for matching the GOAL
Backward Chaining, Forward Chaining
What to do when Goal has multiple Subgoals?
Depth-First Search or Breadth- First Search and **Backtracking**
Cut operator

Inferencing (resolution) Process

- To prove a goal TRUE, inference process must find a **CHAIN** of **reference RULE and/or FACTS** in the database that **connect the goal** to **one or more facts** in the database

…contd… Inferencing (resolution) Process

- If Q is a GOAL, then Q must be found as a fact in the data base or the inferencing process must find a fact P1 and a sequence of propositions P2,P3,..,Pn such that

  P2:-P1

  P3:-P2

  ………..

  Q:-Pn

…contd… Resolution Process

- Complications in Resolution Process
  - Presence of compound RHS in RULES
  - Presence of VARIABLES in RULE of Knowledgebase (Solved by **UNIFICATION**)

VARIABLES IN RULE of KB

- likes(rohan,cake).
- likes(rishi,cake).
- likes(rama,burger).
- likes(rohan,burger).
- friend(X,Y) :- likes(X,Z),likes(Y,Z).

...contd... Inferencing (resolution) Process

- SEARCHING for Facts , requires MATCHING of terms
- Proving a **Subgoal** is called **Satisfying** that Subgoal

Samples:

SIMPLEST QUERY : man(bob)

SOLUTION : Search from **top – down** all the facts in knowledgebase for fact man(bob)

…contd… Inferencing (resolution) Process

- QUERY with variable : man(X)
- Fact, Rule in Knowledgebase

father(bob)

man(X):-father(X)

Requires: Unification to instantiate X to bob temporarily

2 Opposite Approaches for Matching Goal to a Fact in Database

1) **Begin with Facts and Rules** of database and attempt to **find a sequence of matches** that **leads to the Goal** - This is **Bottom-up** or **FORWARD CHAINING**

Alternative

2) **Begin with Goal** and attempt to **find a sequence of matching propositions** that **lead to** some **set of original facts** in the database - This is **Top-down** or **BACKWARD CHAINING**

…contd…….2 Opposite Approaches for Matching Goal to a Fact in Database

- For some problems, **Forward Chaining** from facts and rules to main query is more efficient
- General Rule is that – **Forward chaining** performs better when Facts are more than Rules
- **Backward Chaining** is efficient and preferred, when one or few choices of logical statements to resolve with a goal

Prolog Matching Approach

- Prolog Implementations use **Backward Chaining** for matching (resolving) purspose, because its designers believed that larger class of problems are suitable **for Backward Chaining**

Backward Chaining Example

- Query: man(bob)
- Knowledgebase

    father(bob)

    man(X):-father(X)

Process:

1) Start from Goal man(bob)
2) Knowledgebase is searched from top, and LHS of 2nd proposition is matched through Unification to instantiate X with bob.
3) Match RHS of the second proposition with first proposition

What to do when Goal has **multiple Subgoals**?

Solution SEARCH has commonly 2 techniques

- Depth –First (Used in PROLOG)
  - When there are multiple subgoals, the leftmost subgoal is satisfied first and the next, and then so on to the last one on the right
  - A complete sequence of proposition for first Subgoal is found before going to the others
- Breadth – First (Not used in PROLOG)
  - When there are multiple subgoals, the seraching for solution works parallely on all subgoals

Depth First Search (DFS)

- DFS is selected by Prolog designer because fewer computer resources are required
- When Goal has **multiple Subgoals** BACKTRACKING (next slide) is required

Backtracking

- In the process of satisfying Subgoals of the GOAL with multiple subgoals, a situation may arise when the system fails to show True for a current subgoal it is trying to satisfy (prove true). When this happens, system abandon’s thecurrent subgoal and goes back (**BACKTRACKS**) to previous subgoal it already proved true, and attempts to find a new, alternate solution to it.
- This BACKING up to previous subgoal is called **BACKTRACKING**

…Backtracking

- **Backtracking definition** : Backing up in the goal to reconsider a previous proven subgoal.
- Finding new solution:
  - Start from where the search stopped for previous proven goal.
  - Multiple solutions for same goal result from different instantiations of its variable
  - May require a great deal of time and space because it may have to find all possible proofs to every subgoal

Ordering of Subgoals

- E.g.

  male(X), parent(X,shelly)

  (male(X), X has MANY choices)

  change of order

  parent(X,shelly), male(X)

  (parent(X,shelly), X has only2 choices)

Thus Order is Important for efficiency (deficiency of Prolog)

Cut Operator in Prolog

Cut Operator in Prolog

- The Cut operator can control the search space.
- Cut informs Prolog control system which **choices need not be considered again** during **backtracking** to satisfy a subgoal.
- The Cut operator can be introduced by a predicate ! with no arguments.
- Cut operator freezes the system to the choices made since the rule was selected.
- Effect of the Cut is to prune the search tree or to control backtracking during the derivation process.

Arithmetic Example

Simple Arithmetic

KB6

speed(ford, 100).

speed(chevy, 105).

speed(volvo, 80).

time(ford, 20).

time(chevy, 21).

time(volvo, 24).

Distance(X,Y):-speed(X,Speed),time(X,Time), Y is Speed * Time

Query

distance(chevy, Chevy_Distance)

Instantiates Chevy_Distance with value 2205

Prolog Deficiencies

Roosta Pg 378

4 important drawbacks of Prolog

1) **Subgoal order** determines shape of search space that the control system explores in its resolution process
   - Poor subgoal order may produce a search tree with many branches, and system may fail to find solution, even if one exists.

…contd…4 important drawbacks of Prolog

2) **Order in which facts and rules are employed**, is significant in the sequence of the solution obtained

- This is key difference between logic program and Prolog
- Prolog interpreter has a fixed control strategy

…contd…4 important drawbacks of Prolog

3) Because **Prolog interpreter performs exhaustive depth first search** when trying to unify its variables, program execution can be very inefficient, in terms of search time and memory usage

…contd…4 important drawbacks of Prolog

4**) Occurs Check** problem: When control system Unifies a Variable with a Term, it does not check whether the Variable itself occurs in the term it is being instantiate to.

Practice Problems

(in later Doc)