Unit IV

Logic Programming Paradigm

Introduction

- The *Logical Paradigm* takes a declarative approach to problem-solving.
- Various logical assertions about a situation are made, establishing all known facts.
- Then queries are made.
- The role of the computer becomes maintaining data and logical deduction.

What is a logic ?

- A logic is a language.
- It has syntax and semantics.
- More than a language, it has inference rules.
- Syntax:
  - The rules about how to form formulas;
  - This is usually the easy part of a logic.
- Semantics:
  - about the meaning carried by the formulas,
  - mainly in terms of logical consequences.
- Inference rules
  - Inference rules describe correct ways to derive

Features of Logical paradigms

- Computing takes place over the domain of all terms defined over a "universal" alphabet.
- Values are assigned to variables by means of automatically generated substitutions, called most general unifiers.
- These values may contain variables, called logical variables.
- The control is provided by a single mechanism: automatic backtracking.
- Strength - simplicity and Conciseness
- Weakness - has to do with the restrictions to one control mechanism and the use of a single data type.

Logical Paradigm Programming

- **Logic programming** is a computer programming paradigm where program statements express facts and rules about problems within a system of formal logic.
- Rules are written as logical clauses with a head and a body; for instance, "H is true if B1, B2, and B3 are true."
- Facts are expressed similar to rules, but without a body; for instance, "H is true.“
- Languages used for logic programming : Datalog, Prolog (PROgramming in LOGic), Alice, ASP(Answer Set Programming)

Logical Paradigm Programming

- A logical program is divided into three sections:
  - a series of definitions/declarations that define the problem domain
  - statements of relevant facts
  - statement of goals in the form of a query
- Any deducible solution to a query is returned.
- The definitions and declarations are constructed entirely from relations. i.e. X is a member of Y or X is in the internal between a and b etc.

Advantages

- The advantages of logic oriented programming are bifold:
  - The system solves the problem, so the programming steps themselves are kept to a minimum
  - Proving the validity of a given program is simple.

History of Logic Programming (LP)

- Logic Programming has roots going back to early AI researchers like John McCarthy in the 50s & 60s
- Alain Colmerauer (France) designed Prolog as the first LP language in the early *1970s*
- Bob Kowalski and colleagues in the UK evolved the language to its current form in the late 70s
- It's been widely used for many AI systems, but also for systems that need a fast, efficient and clean rule based engine
- The prolog model has also influenced the database community
  - Exp datalog

Logic Programming Paradigm Example

- domains
  - being = symbol
- Predicates
  - animal(being) % all animals are beings
  - dog(being) % all dogs are beings
  - die(being) % all beings die
- Clauses
  - animal(X) :- dog(X) % all dogs are animals
  - dog(fido). % fido is a dog
  - die(X) :- animal(X) % all animals die

Python Logic Programming

- Python Logic programming is a programming paradigm that sees computation as automatic reasoning over a database of knowledge made of facts and rules.
- It is a way of programming and is based on formal logic.
- A program in such a language is a set of sentences, in logical form, one that expresses facts and rules about a problem domain.
- Among others, Datalog is one such major logic programming language family.

Structure of Python Logic Programming

- Facts are true statements
  - Example :Bucharest is the capital of Romania.
- Rules are constraints that lead us to conclusions about the problem domain. These are logical clauses that express facts.
- Syntax to write a rule (as a clause):

  H :- B1, …, Bn.

- We can read this as:
  H if B1 and … and Bn.
  Here, H is the head of the rule and B1, …, Bn is the body.
- A fact is a rule with no body: **H.**

Structure of Python Logic Programming

- Example

  fallible(X) :- human(X)
- Every logic program needs facts based on which to achieve the given goal.
- Rules are constraints that get us to conclusions.

Structure of Python Logic Programming

- Logic and Control
- An algorithm as a combination of logic and control. **Algorithm=Logic+Control**
- In a pure logic programming language, the logic component gets to the solution alone.
- We can, however, vary the control component for other ways to execute a logic program.

Python Logic Programming

- Install a couple of **packages**. Let's use pip for this.
- **Kanren-** It lets us express logic as rules and facts and simplifies making code for business logic.

  >>> pip install kanren
- **SymPy-** This is a Python library for symbolic mathematics. It is nearly a full-featured Computer Algebra System.

  >>> pip install sympy

Example1

- With logic programming, we can compare expressions and find out unknown values.

```
from kanren import run,var,fact
from kanren.assoccomm import eq_assoccomm as eq
from kanren.assoccomm import commutative,associative
add='add' #Defining operations
mul='mul‘
fact(commutative,mul) #Addition and multiplication are commutative and associative
fact(commutative,add)
fact(associative,mul)
fact(associative,add)
a,b,c=var('a'),var('b'),var('c') #Defining variables
#2ab+b+3c is the expression we have'
expression=(add, (mul, 2, a, b), b, (mul, 3, c))
expression=(add,(mul,3,-2),(mul,(add,1,(mul,2,3)),-1)) #Expression
expr1=(add,(mul,(add,1,(mul,2,a)),b),(mul,3,c)) #Expressions to match
expr2=(add,(mul,c,3),(mul,b,(add,(mul,2,a),1)))
expr3=(add,(add,(mul,(mul,2,a),b),b),(mul,3,c))
run(0,(a,b,c),eq(expr1,expression)) #Calls to run()
```

- Output

  ((3, -1, -2),)
- >>> run(0,(a,b,c),eq(expr2,expression))

  ((3, -1, -2),)
- >>> run(0,(a,b,c),eq(expr3,expression))

  ()

Example 2

- Checking for Prime Numbers in Python Logic Programming

```
from kanren import isvar,run,membero
 from kanren.core import success,fail,goaleval,condeseq,eq,var
from sympy.ntheory.generate import prime,isprime
import itertools as it

def prime_test(n): #Function to test for prime
    if isvar(n):
            return condeseq([(eq,n,p)] for p in map(prime,it.count(1)))
    else:
            return success if isprime(n) else fail
n=var()                                         #Variable to use
set(run(0,n,(membero,n,(12,14,15,19,21,20,22,29,23,30,41,44,62,52,65,85)),(prime_test,n)))
```

- **Output: {41, 19, 29, 23}**
- >>> run(7,n,prime_test(n))

  (2, 3, 5, 7, 11, 13, 17)

Datalog Concepts

- pyDatalog is a powerful language with very few syntactic elements, mostly coming from Python :
- Variables and expressions
- Loops
- Facts
- Logic Functions and dictionaries
- Aggregate functions
- Literals and sets
- Tree, graphs and recursive algorithms
- 8-queen problem

Reference

- https://sites.google.com/site/pydatalog/Online-datalog-tutorial

PySwip Introduction

- PySwip is a Python - SWI-Prolog bridge enabling to query SWI-Prolog in your Python programs.
- It features an (incomplete) SWI-Prolog foreign language interface, a utility class that makes it easy querying with Prolog and also a Pythonic interface.
- Since PySwip uses SWI-Prolog as a shared library and ctypes to access it,
- it doesn't require compilation to be installed.

- Reference :
  - https://pypi.org/project/pyswip/