CSE 425 / L-25 / 25.11.2024 /

Python - Practice
Jupyter Notebook

Quize-2
Next Class
upto List

L-26 / 27.11.2024 /

Quiz-2

Logic Programming

⇒ Facts, rules
- Facts → information
- rules → Defined by the user
  - inference rules ⇒ Path
- logic
  ⇒ predicate calculus

Logic:

- Declarative ⇒ semantics
- has to do with propositional form → predicate

Propositional Form:

⇒ Declarative statement that is definitely convey truth values → True / False

P: Proposition, Q: Proposition } Merge Compound Proposition } we use logical operators } ∧: Conjunction, ∨: Disjunction, →: Implication

p	q	p → q
T	T	T
T	F	F
F	T	T
F	F	T

→ Vacously True

Predicate!

⇒ Every student in CSE 425 has done CSE 173.

Predicate Definition:

P(x) : Student x from CSE 425 has done CSE 173

x ∈ Domain
↳ CSE 425

⇒ True/False ?

- process to obtain is Quantification
  - ↳ Quantifier
    - ∀ (every, none, nobody, all) → Universal
    - ∃ → Existential
      - ↳ some, a few, at least a few, exactly 3

∀x P(x) ⇒ True/False ?

$P(x=a) \wedge P(x=b) \wedge \cdots \wedge P(x=z)$

∃x P(x)

P(x) : x owns a GPU

⇒ there exist at least one x such that P(x)

$P(x=a) \vee P(x=b) \vee \cdots \vee P(x=z)$

L-27/02.12.2024/

Theory:

- Logic Programming → Propositional Algebra
  - Predicate Calculus
  - Inference
    - ⇒ Rules
      - resolution
        - resolution principle
          - used to resolve queries in logic programming } Backward Chaining

$x \in$ Domain

$\forall x\ P(x)$: For all x $P(x)$

$\exists x\ P(x)$: There exist at least one x such that $P(x)$

⇒

$x \in NSU$

$P(x)$: x is from SEC-1

$Q(x)$: x is good

⇒ $\forall x\ (P(x) \rightarrow Q(x))$

⇒ $x \in NSU$

$P(x)$: x from SEC-1

$S(x)$: x owns a super computer

$\exists x\ P(x) \wedge S(x)$

Inference:

- to infer
- to guess based on available information & logical reasonings / deductions.
- we have rules
  - ↳ are in propositional form.

Rules:

(i) $p \to q$, p / $\therefore q$

(ii) $p \to q$, $\neg q$ / $\therefore \neg p$

(iii) $p \wedge q$ / $\therefore p, q$

(iv) p, q / $\therefore p \wedge q$

(v) p ...] True / $\therefore p \vee q \vee r \vee s \vee \ldots$

(vi) Hypothetical Syllogism

$p \to q$

$q \to r$

$\therefore p \to r$

(vii) Disjunction Syllogism

$p \vee q$

$\neg p$

$\therefore q$

(viii) Resolution → Clausal Form

$p \vee q$] True

$\neg p \vee r$

$\therefore q \vee r$

Clausal Form:

- disjunction of literals or the single literal itself.
- Extracted form
  - Conjunctive Normal Form (CNF) ⇒ $\underbrace{(\ldots \vee \ldots)}_{\text{clausal form}} \wedge (\ldots \vee \ldots) \ldots$

Resolution Principle

$\neg P \rightarrow (Q \wedge R)$

$\equiv \neg(\neg P) \vee (Q \wedge R)$ → Definition of implication

$\equiv P \vee (Q \wedge R)$

$\equiv \underbrace{(P \vee Q)}_{\text{Clause-1}} \wedge \underbrace{(P \vee R)}_{\text{Clause-2}}$

Resolvent:

$\Rightarrow C_1 = P \vee Q \vee R : C_1' = Q \vee R$

$\Rightarrow C_2 = \neg P \vee \neg S \vee T : C_2' = \neg S \vee T$

$\vee \Rightarrow \underbrace{C_1' \vee C_2'}_{\text{Resolvent}}$

1. $T \rightarrow (M \vee E) \rightarrow \neg T \vee M \vee E$
2. $S \rightarrow \neg E \rightarrow \neg S \vee \neg E$
3. $T \wedge S \rightarrow T$, S } Clausal itself

$\therefore M$ → Desired Conclusion

$\boxed{\neg M}$ → adding negated conclusion as a clausal form

1. $\neg T \vee M \vee E$
2. $\neg S \vee \neg E$
3. T
4. S
5. $\neg M$ → Now calculate the resolvent?
6. $\neg T \vee M \vee \neg S$ ∵ Resolvent 1, 2
7. $M \vee \neg S$ ∵ Resolvent 3, 6
8. M ∵ Resolvent 7, 6
9. $\square$ ∵ Resolvent 8, 5 → By definition, it is a FALSE FLAG

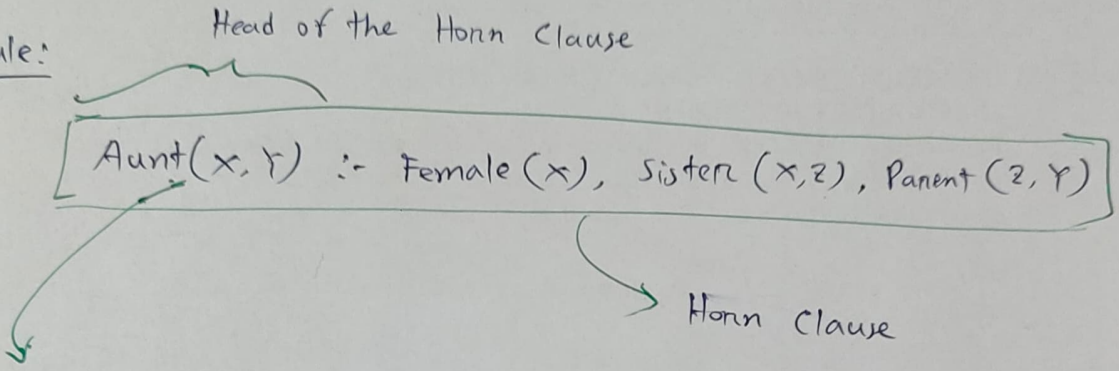
∴ So conclusion M is True (Proved by Contradiction)

✳ Example:

Sister (x, z) : x is a sister of z

Parent (z, y) : z is the parent of y

Rule:



✳

$\forall x \forall y \forall z \; (Female(x) \wedge Sister(x,z) \wedge Parent(z,y))$

Definition of Aunt

Transform it to proposition

Instantiation Process

$x \in$ Domain

$\forall x \; P(x)$] True

$\therefore P(c)$

Specific subject

L-28 / 03.12.2024 /

Proposition Topic from 173 must read

* Prolog knowledge Base
  - Facts (Predicate)
  - Rules: Defined as Horn Clause

"to be define" :- "Predicate Calculus"

↳ sequence of predicate

|||

"sequence of predicate" → "term to be defined."

* Previous Example:

Female (Sita)

Sister (Sita, Geetha)

Parents (Geetha, Mohan)

Horn clause with Head: Rule : Aunt (X,Y) :- Female (X), Sister (X,Z), Parents (Z,Y)

X is sister of Z

↳ X is the Aunt of Y

↳ Z is the parent of Y

"," ≡ AND ≡ Conjunction ≡ ∧

";" ≡ OR ≡ Disjunction ≡ ∨

* $\forall X \forall Y \forall Z \Big(\big(Female(X), Sister(X,Z), Parent(Z,Y) \big) \rightarrow Aunt(X,Y) \Big)$

∴ Aunt (Sita, Mohan) = ?

⇒

1. Female (Sita)
2. Sisten (Sita, Geetha)
3. Parent (Geetha, Mohan)

} Clause.
- each as propositional form : subject is sep specified.

✱ Taking instantiation of the rules

Universal

$\forall x\, P(x)$] True

$\therefore P(c)$

↳ Specific subject

Existential

$\exists x\, P(x)$] True

$\therefore P(a)$

4. $(\text{Female}(\text{Sita}) \wedge \text{Sisten}(\text{Sita}, \text{Geetha}) \wedge \text{Parent}(\text{Geetha}, \text{Mohan})) \rightarrow \text{Aunt}(\text{Sita}, \text{Mohan}))$

$\equiv \neg(\text{Female}(\text{Sita}) \wedge \text{Sisten}(\text{Sita}, \text{Geetha}) \wedge \text{Parent}(\text{Geetha}, \text{Mohan})) \vee \text{Aunt}(\text{Sita}, \text{Mohan})$

$\equiv \neg \text{Female}(\text{Sita}) \vee \neg \text{Sisten}(\text{Sita}, \text{Geetha}) \vee \neg \text{Parent}(\text{Geetha}, \text{Mohan}) \vee \text{Aunt}(\text{Sita}, \text{Mohan})$

5. $\neg \text{Aunt}(\text{Sita}, \text{Mohan})$

6. $\square \Rightarrow$ [Line 4 and 1, 2, 3, 5 all are eliminated as opposite form and both are True.]

↳ Empty Clause

↳ FALSE

Practice on Swish Prolog

↳ always use trace to understand the step.

Question Pattern:

- Graph will be given
  - define edge on connected lines, (visual graph)
  - ~~wright~~ ,
  - write the rule and proposition
  - need to write all edge predicate, undirect ⇒ add both way

⇒ edge graph
⇒ Mathematical function (area, square, factorial, Fibonacci)
⇒ Relationship - family tree
⇒ List Example (May not important for exam)

L-29/04.12.2024/

λambda Calculus:

function ⇒ Prefix of a statement.

Functional Programming Language (FPL):

OCaml PL

- Handle symbolic computation
- List processing application
  ↳ flexible storage of values.
- we are concerned more on the functionality & less about "memory related" variable storage.

⇒ Two sub-class

Pure FPL
- Haskell
- Supports only functional paradigm

Impure FPL → Von Neuman Hardware
- Allows imperative system as well
- LISP

Applications
- mainly AI area
- Natural Language Processing
- Speech and vission processing.

Example:

C

```
sum = 0;
for (i=1; i<=n; i++){
    sum = sum+i;
}
```

```
x = 7;
```

FPL

```
sum [1,2,3,...,n]
```
→ function does the computation.

```
x = f(y)
```

Functional Programming Language ⇒ Elements
- List
- Function
- Composition (gof, fog)
- Recursive

Lambda Calculus:

<expression> → <name> | <function> | <application>

<function> → λ <name> . <expression>

<application> → <expression> <expression>

E E

$E_1, E_2, E_3, \ldots E_n$

( ) ⇒ for separating expression.

(+ 3 4) ⇒ 3+4 = 7

((+ 3) 4) ⇒ 3+4 = 7 (as input of)

λx. x * x

bound variable

function of x

expression

Left association:

$((((E_1)\ E_2)\ E_3) \ldots E_n)$

Free variable:

(λx. x)

Body
- expression
  - x
  - bound

(λy. yx)

body
- expression
  - yx
    - Bound
    - Free

(+ (* 5 6) (* 8 3))

→ (+ (30) (* 8 3))

→ (+ 30 24)

→ 54

(*) $(\lambda x(\lambda y.\ x+y)\ 5)\ ((\lambda y.\ y*y)\ 6)$ = ?

→ $(\lambda x.\ x+5)\ ((\lambda y.\ y*y)\ 6)$

→ $(\lambda x.\ x+5)\ (6*6)$

→ $(6*6)+5$

→ 41.

One Question must from Lambda function

Map function.

Final Exam - 19.12.2024