# Introduction to Logic Programming

## Programming Language Paradigms

- **Imperative programming**
  - Semantics: state based
  - Computation viewed as state transition process
  - Categories:
    - Procedural
    - Object Oriented
    - Other non-structured
- For example: C, Pascal, Turing are in the Procedural category, steps of computation describe state changing process



## Programming Language Paradigms…

- **Declarative Programming**
  - Focus is on logic (WHAT) rather than control (HOW)
  - Categories:
    - Logic Programming: Computation is a reasoning process, e.g. Prolog
    - Functional Programming: Computation is the evaluation of a function, e.g. Lisp, Scheme, …
    - Constrained Languages: Computation is viewed as constraint satisfaction problem, e.g. Prolog (R)



## Programming Language Paradigms…

**Procedural programming**

- This is a programming style in which the programmer specifies in detail how the programming problems are solved
- This style of programming is supported by third generation languages such as PASCAL, C ,FORTRAN, BASIC, COBOL
- This programming style can also be used to support traditional AI languages such as PROLOG and LISP

## Programming Language Paradigms…

**Functional programming**

- This is a programming technique that is dominated by reliance on elements that call the in-built function primitives
- Problem solutions are formulated as a series of function calls
- LISP and its variants form a popular environment for using functional programming to solve programming problems
- Typical format is (function, arguments.)



## Programming Language Paradigms…

**Object-oriented programming**

- This is a style of programming in which object oriented paradigm is supported
- Objects and their behavior can be defined and manipulated
- There are specialized languages supporting object oriented programming such as Smalltalk, C++ and Object PASCAL
- Object-oriented programming support can be used in some AI programming tasks



## Programming Language Paradigms…

**Declarative programming**

- This is a programming style in which only what is required is specified without concern to how the operations are carried out
- This kind of programming is supported by traditional AI languages such as PROLOG and LISP
- It is also supported by fourth generation languages such as form builders of relational databases



## Propositional Logic

- A proposition is a statement of language that formulates something about an external world.
- A proposition can either be true or false.
- Questions, commands, promises, etc., are not propositions.

## Propositional Logic…

- Propositional logic consists of symbols that represent whole propositions (facts)
- For example **B** may represent **'It is sunny outside',** that may be **true** or **false**.
- Propositions may be combined using Boolean connectives which generate sentences with more complex meanings
- Little commitment is made to how things are represented thus limiting propositional logic as representation language
- A statement can be used to derive new propositions of conclusions



## Propositional Logic…

- It is usual to use symbols, such as letters of the alphabet or abbreviations, to represent various propositions, or conclusions
- One can use logical connectives such as AND, OR, NOT, IMPLIES, EQUIVALENT
- Propositional Logic deals with complete statements and whether they are true or false



## Propositional Logic…

- We express statements as a whole, and combinations of them
- A statement is a sentence in which something is told about some reality, and which can be true or false about that reality.
- For example, if p is the statement "Albert is at home", and q means "the door is locked", then $q \rightarrow \neg p$ says: "if the door is locked, then Albert is not at home".



## Propositional Logic…

**Syntax of propositional logic**

- Proposition symbols: capital letters eg. P, Q
- Logical constants: True/ False
- Logical connectives: $\wedge$(and), $\vee$(or), $\Leftrightarrow$ (equivalent), $\Rightarrow$ (implies), $\neg$ **(NOT)**, and ()

## Propositional Logic…

**Syntax of propositional logic….**

- Rules for putting the sentences together:
  - Logical constants True and False are sentences by themselves
  - Propositional symbol such as P or Q are sentences by themselves
  - Enclosing a sentence by parentheses yield a sentence, e.g. (P∧Q).
  - A sentence can be formed, by combining sentences by any one on the five logical connectives



## Propositional Logic…

- ∧ (and). A sentence whose connective is ∧, such as P∧(Q∨R), is called a ***conjuction***(logic) and its parts are called ***conjuncts.***
- ∨ (or). A sentence using ∨, such as A∨(P∧Q) is called a ***disjunction*** of ***disjunts***.
- ⇒(implies). A sentence such as (P∧Q) ⇒ R is called ***implication*** (or conditional). Its ***premise*** or ***antecedent*** is (P∧Q) and its ***conclusion*** is R. Implications are also known as ***rules*** or ***if-then*** statements.
- ⇔(equivalent). The sentence (P∧Q) ⇔ (Q∧P) is an ***equivalence*** (also called ***biconditional***).



## Propositional Logic…

- ¬ **NOT/Negation**
- Order of precedence in propositional logic is: ¬, ∧, ∨, ⇒, ⇔



## Propositional Logic…

***Semantics of propositional logic***

- Concerned with specifying the interpretations of the proposition symbols and the meanings of the logical connectives
- Proposition symbol: this can be any thing that is desired. A symbol can stand for any arbitrary fact
- Logical constants: **True** corresponds to the way the world is, that it is actually a fact. **False** corresponds to the way the world is not, that the world is not

## Propositional Logic…

- ***Semantics of propositional logic…***
- A complex sentence derives meaning from its parts
- Connectives can be considered as functions that take in truth values (True or False) and return a truth value
- The truth table below gives the interpretations of connectives

| P | Q | ¬P | P∧Q | P∨Q | P⇒Q | P⇔Q |
|---|---|---|---|---|---|---|
| False | False | True | False | False | True | True |
| False | True | True | False | True | True | False |
| True | False | False | False | True | False | False |
| True | True | False | True | True | True | True |



## Propositional Logic…

- The implication, P⇒Q, is also equivalent to ¬P ∨Q.
- Consider a complex sentence such as: (P∨Q) ∧ ¬S. This sentence is decomposed first into (P∨Q) and ¬S, the meanings of each is determined independently. The result of each is then combined by the ∧.



## Propositional Logic…

***Validity***

- A sentence of the form: Premises ⇒ Conclusion is valid if all the premises of ⇒ are all true. A truth table can be used to check if a sentence is valid.
- Exercise check that ((P∨H)∧¬H)⇒ P is valid.
- Validity is a way that enables the machine to check premises to determine if the conclusion is true. The process of determining if a sentence if true is called *inference*. The machine can perform inference by building truth tables for the sentence and checking that in the sentence Premises ⇒ Conclusion, that the row is all true for the ⇒.



## Propositional Logic…

***Rules of inference for propositional logic***

- Patterns of inferences are captured in rules. The rules enable inference to be made without building truth tables. Consider the following notations:-
- A |-B, means that B can be derived from A by inference. An inference rule is sound if the conclusion is true in all cases where the premises are true.
- The following are a list of seven most commonly used inference rules.

## Propositional Logic…

- **Modus ponens (or implication-Elimination):** given an implication and the premise, you can infer the conclusion.
  - $A \Rightarrow B, A \mid\!- B$.
- **AND-Elimination:** given a conjunction, you can infer any conjunct.
  - $A_1 \wedge A_2 \wedge A_3 \wedge \ldots\ldots \wedge A_n \mid\!- A_i$
- **AND-Introduction:** given a list of sentences, you can infer their conjunction.
  - $A_1, A_2, A_3, \ldots\ldots, A_n \mid\!- A_1 \wedge A_2 \wedge A_3 \wedge \ldots\ldots \wedge A_n$
- **OR-Introduction**: given a sentence, you can infer all its disjunctions with any thing else.
  - $A_I \mid\!- A_1 \vee A_2 \vee A_3 \vee \ldots\ldots \vee A_n$



## Propositional Logic…

- **Double-Negation Elimination:** given a doubly negated sentence infer a positive sentence.
  - $\neg\neg A \mid\!- A$
- **Unit resolution:** given a disjunction, if one of the disjuncts is false then infer the other. This is a special case of resolution.
  - $A \vee B, \neg B \mid\!- A$, alternatively: $A \vee B, \neg A \mid\!- B$
- **Resolution:** implication is transitive.
  - $A \vee B, \neg B \vee C \mid\!- A \vee C$ or equivalently: $\neg A \Rightarrow B, B \Rightarrow C \mid\!- \neg A \Rightarrow C$.



## Propositional Logic…

**Monotonicity**

- Monotonicity occurs where adding rules does not change the status of some other rules. Consider a knowledge base KB, that entails some sentences. Now suppose that some sentences are added. Then the sentences entailed in KB are also entailed in the new knowledge base. This may also be expressed as:
  - If $KB_1 \mid\!- A$ then $(KB_1 \cup KB_2) \mid\!- A$.

**Horn sentences (clauses)**

- A Horn sentence is a sentence in the form: $A_1 \wedge A_2 \wedge A_3 \wedge \ldots\ldots \wedge A_n \Rightarrow B$



## First order Predicate Logic

- ***Predicate logic*** or first order logic, provides a way of representing the world in terms of objects and predicates on objects (property or relation between objects).
- The ***connectives*** are used. ***Quantifiers*** which are statements that allow description about the universe at once are also used
- A statement has a specific inner structure, consisting of terms and predicates.
- Terms denote objects in some reality, and predicates express properties of, or relations between those objects

## First order Predicate Logic…

- In predicate logic, one can break a statement into component parts.
- These parts are either:
  - Object
  - Characteristic
  - Relationship
  - Some assertion about the object.



## First order Predicate Logic…

- In addition, predicate logic lets us use variables in a symbolic logic statement
- Example 1:
  - Jane is naughty. She combs her hair in class
    - is (jane, naughty).
    - combs (jane, hair_class).



## First order Predicate Logic…

- Example 2:
  - Mwangi travels by matatu, bus, bicycle.
    - travels (mwangi, matatu)
    - travels (mwangi, bus).
    - travels (mwangi, bicycle).
    - travels (mwangi, X).
    - X= {matatu, bus, bicycle}



## First order Predicate Logic…

- **Propositional logic** is limited in that the component propositions cannot be decomposed and be individually examined. The only ontological commitment is that the world consists of facts. This is limiting in representation or description of the world
- **Predicate logic** is an extension of propositional calculus. Its main ontological commitment is that the world consists of objects which are individual things and that these objects have properties that distinguish them from other objects
- Objects are related by relations, and some of the relations are functions giving only one value for a given input

## First order Predicate Logic…

***Example***

- ***Objects:*** people, houses, numbers, theories, James Kinyanjui, colors, soccer games, wars, centuries, …..
- **Relations:** *brother of, bigger than, inside, part of, has, color, occurred after, owns, …*
- ***Properties:*** red, round, bogus, prime, multistoried, ….
- ***Functions:*** father of, best friend, one more than, ….
- Note that any fact can be referred to as an object, property or relation. Also note that predicate calculus has no ontological commitments to categories, events or time.



## First order Predicate Logic…

**Example 1**

IF there is a lot of oil demand in China THEN global oil prices rise.

(china_oil_demand, a_lot) → will(global_oil_price, rise).

***In Prolog***
is (china_oil_demand, a_lot) :- will(global_oil_price, rise).



## First order Predicate Logic…

**Example 2**

Either Kenya, Uganda or Tanzania will join the U.N. security council.

(kenya ∨ uganda ∨ tanzania)→ join_council

***In Prolog***
will( join_council , kenya) ; will ( join_council, uganda) ; will( join_council, tanzania).



## First order Predicate Logic…

**Example 3**

For Inflation to occur there must be bad export business, poor economic management and weak currency.

bad_export ∧ poor_economic_management ∧ weak_currency → inflation

***In Prolog***
(bad_export , poor_economic_management , weak_currency) :- inflation

## Logic Programming

- Logic programming is a programming paradigm based on mathematical logic
- In this paradigm the programmer specifies relationships among data values (this constitutes a logic program) and then poses queries to the execution environment (usually an interactive interpreter) in order to see whether certain relationships hold
- A logic program, through explicit facts and rules, defines a base of knowledge from which implicit knowledge can be extracted



## Logic Programming…

- Logic programming is popular for data base interfaces, expert systems, mathematical theorem provers, heuristic searches, and constraint satisfaction problems.
- A functional program consists of a sequence of function definitions. A logic program consists of a sequence of relation definitions
- Both functional and logic programs rely heavily on recursive definitions
- The big difference between functional and logic programs is in the underlying execution "engine", that is, the imperative parts of the languages



## Logic Programming…

- The execution engine of a functional language evaluates an expression by converting it to an acyclic graph and then reducing the graph to a normal form which represents the computed value
- The *Prolog* execution environment, on the other hand, doesn't so much "compute" an answer, it "deduces" an answer from the relation definitions at hand. Rather than being given an expression to evaluate



## END