Predicate Logic

The propositional logic, is not powerful enough for all types of assertions;

Example : The assertion "**x > 1**", where **x** is a variable, is not a proposition, because ;

it is neither true nor false unless value of **x** is defined.

For **x > 1** to be a proposition ,

either we substitute a specific number for **x** ;

or change it to something like

"There is a number x for which x > 1 holds";

or **"For every number x, x > 1 holds"**.

Consider example :

“ All men are mortal.

Socrates is a man.

Then Socrates is mortal” ,

- These cannot be expressed in propositional logic as a finite and logically valid argument (formula).
- We need languages : that allow us to describe properties (***predicates***) of objects, or a relationship among objects represented by the variables .
- Predicate logic satisfies the requirements of a language.
- *Predicate logic* is powerful enough for expression and reasoning.
- *Predicate logic* is built upon the ideas of *propositional logic.*

- ■ **Predicate :**
- Every complete "sentence" contains two parts : a “Object" and a “Predicate".
- The *Object* is what (or whom) the sentence is about.
- The *Predicate* tells something about the Object;
- **Example :**
- A *sentence* **"Judy {runs}".**
- The Object is **Judy** and the predicate is **runs** .
- Predicate, always includes verb, tells something about the Object.
- **Predicate is a verb phrase template that describes a property of objects, or a relation among objects represented by the variables.**

- **Example:**

  “The car Tom is driving *is blue*" ;

  "The sky *is blue*" ;

  "The cover of this book *is blue*“

- Predicate is **“is blue"** , describes property.
- Predicates are given names; Let **‘B’** is name for predicate **"is_blue"**.
- Sentence is represented as **"B(x)"** , read as **"x is blue"**;
- Symbol **“x”** represents an arbitrary Object .

- Definition: A predicate is a property that a variable or a finite collection of variables can have.
- A predicate becomes a proposition when specific values are assigned to the variables.
- P(x1, x2, ..., xn) is called a predicate of n variables or n arguments.
- Example:

  He lives in the city.
- Predicate : P(x,y): x lives in y.
- P(Aly, Cairo) is a proposition: Aly lives in Cairo.

▪ Predicate Logic Expressions :

- The propositional operators combine predicates, like
  If (p(....) && (!q(....) || r (....)))

Logic operators :

- Examples of disjunction (**OR**) and conjunction (**AND**).
- Consider the expression with the respective logic symbols **||** and **&&**
  x < y || (y < z && z < x) which is
  true || (true && true) ;
- Applying truth table, found **True**
- Assignment for **<** are **3, 2, 1** for **x, y, z** and then the value can be *FALSE* or *TRUE*
- **3 < 2 || (2 < 1 && 1 < 3)** It is False

Predicate Logic Quantifiers

- As said before, **x>1** is not proposition
- Quantify the variable using a quantifier on formulas of predicate logic (called **wff** well-formed formula), such as **x > 1** or **P(x)**, by using Quantifiers on variables.

- **Apply Quantifiers on Variables**

‡ **Variable x**

* **x > 5** is not a proposition, its truth depends upon the value of variable **x**

* to reason such statements, **x** need to be declared

‡ **Declaration** $x : a$

* $x : a$ declares variable x
* $x : a$ read as **"x is an element of set a"**

‡ **Statement** p is a statement about x

* $Q\ \underbrace{x : a}\ \bullet\ p$ is quantification of statement

statement

declaration of variable x as element of set a

quantifier

* Quantifiers are two types :

universal quantifiers , denoted by symbol $\forall$ and

existential quantifiers , denoted by symbol $\exists$

■ Universe of Discourse

- The universe of discourse, also called domain of discourse or universe.

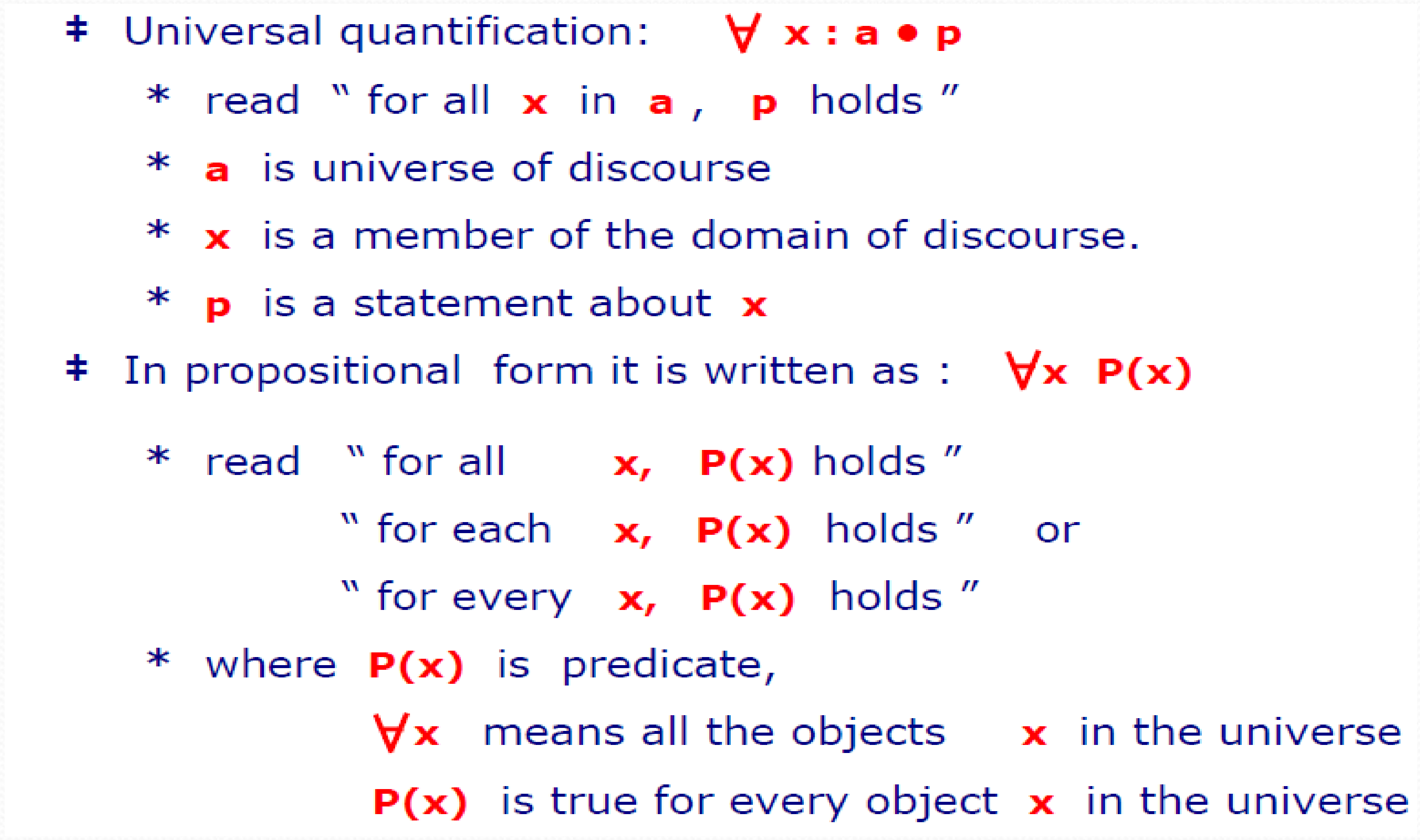
This indicates :

  - a *set of entities* that the quantifiers deal.
  - *entities* can be set of real numbers, set of integers, set of all cars on a parking lot, the set of all students in a classroom etc.
  - *universe* is thus the domain of the (individual) variables.
  - *propositions* in the predicate logic are statements on objects of a universe**.**

- The universe is often left implicit in practice, but it should be obvious from the context.

■ Apply Universal Quantifier " For All "

- Universal Quantification allows us to make a statement about a collection of objects.



‡ Universal quantification: $\forall$ x : a • p

- * read " for all x in a , p holds "
- * a is universe of discourse
- * x is a member of the domain of discourse.
- * p is a statement about x

‡ In propositional form it is written as : $\forall$x P(x)

- * read " for all x, P(x) holds "
  " for each x, P(x) holds " or
  " for every x, P(x) holds "
- * where P(x) is predicate,
  $\forall$x means all the objects x in the universe
  P(x) is true for every object x in the universe

‡ <u>Example</u> : English language to Propositional form

* **"All cars have wheels"**

  $\forall$x : car • x has wheel

* **x P(x)**

  where **P (x)** is predicate tells : **'x has wheels'**

  x is variable for object **'cars'** that populate universe of discourse

■ Apply Existential Quantifier $\exists$ " There Exists "

Existential Quantification allows us to state that an object does exist without naming it.

- ‡ Existential quantification: $\exists$ **x : a • p**
  - * read " there exists an **x** such that **p** holds "
  - * **a** is universe of discourse
  - * **x** is a member of the domain of discourse.
  - * **p** is a statement about **x**
- ‡ In propositional form it is written as : $\exists$ **x P(x)**
  - * read " there exists an **x** such that **P(x)** " or " there exists at least one **x** such that **P(x)** "
  - * Where **P(x)** is predicate

    $\exists$ **x** means at least one object **x** in the universe

    P(x) is true for least one object **x** in the universe

- Example : English language to Propositional form

  - " Someone loves you "

    ∃ x : Someone • x loves you

  - x P(x)

    where P(x) is predicate tells : ' x loves you '

    x is variable for object ' someone ' that populate universe of discourse

3. KR Using Rules

- The other popular approaches to Knowledge representation are called *production rules* , *semantic net* and *frames*.
- **Production rules**, sometimes called **IF-THEN** rules are most popular KR.
- production rules are simple but powerful forms of KR.
- production rules provide the flexibility of combining declarative and procedural representation for using them in a unified form.
- **Examples** of production rules :
  - IF condition THEN action
  - IF premise THEN conclusion
  - IF proposition p1 and proposition p2 are true THEN proposition p3 is true

Types of Rules

- Three types of rules are mostly used in the Rule-based production systems.

■ **Knowledge Declarative Rules :**

- These rules state all the **facts** and relationships about a problem.
- **Example** :

  IF inflation rate declines

  THEN the price of gold goes down.

- These rules constitute a part of the knowledge base.

■ Inference Procedural Rules

- These rules advise on how to solve a problem, while certain facts are known.
- Example :

  IF the data needed is not in the system

  THEN request it from the user.

- These rules are part of the inference engine.

- ■ **Meta rules**
- These are rules for making rules.
- Meta-rules reason about which rules should be considered for firing.
- Example :

  IF the rules which do mention the current goal in their premise,

  AND there are rules which do not mention the current goal in their premise,

  THEN the former rule should be used in preference to the latter.

- – Meta-rules direct reasoning rather than actually performing reasoning.
- – Meta-rules specify which rules should be considered and in which order they should be invoked.

3.1 Procedural versus Declarative Knowledge

■ **Procedural Knowledge : knowing 'how to do'**

- Includes : *rules, strategies, agendas, procedures, models*.
- These explains *what to do* in order to reach a certain conclusion.
- Example:
- Rule: To determine if Peter or Robert is older, first find their ages.
- It is knowledge about *'how to do*' something.
- Accepts a description of the steps of a task or procedure.
- It Looks similar to declarative knowledge, except that tasks or methods are being described instead of facts or things.

■ **Declarative Knowledge : knowing 'what', knowing 'that'**

Includes : *concepts*, *objects*, *facts*, *propositions*, *assertions*, *models.*

It is knowledge about *facts* and *relationships*, *that*

– can be expressed in simple and clear statements,

– can be added and modified without difficulty.

- Examples :
  - A car has four tires;
  - Peter is older than Robert.

- Declarative knowledge and explicit knowledge are articulated knowledge and may be treated as synonyms for most practical purposes.
- Declarative knowledge is represented in a format that can be manipulated, decomposed and analyzed independent of its content.

3.2 Logic Programming

- Logic programming offers a formalism for specifying a computation in terms of logical relations between entities.

– logic program is a collection of logic statements.

– programmer describes all relevant logical relationships between the various entities.

– computation determines whether or not, a particular conclusion follows from those logical statements.

- **Characteristics of Logic program**
- Logic program is characterized by set of relations and inferences.

– program consists of a set of axioms and a goal statement.

– rules of inference determine whether the axioms are sufficient to ensure the truth of the goal statement.

– execution of a logic program corresponds to the construction of a proof of the goal statement from the axioms.

– programmer specify basic logical relationships, does not specify the manner in which inference rules are applied.

Thus **Logic + Control = Algorithms**

Examples of Logic Statements

– Statement:

A grand-parent is a parent of a parent.

– Statement expressed in more closely related logic terms as:

A person is a grand-parent if she/he has a child and that child is a parent.

– Statement expressed in first order logic as

(for all) x: grandparent (x, y) :- parent (x, z), parent (z, y)

read as:

x is the grandparent of y if x is a parent of z and z is a parent of y

Example 1 :

"cows eat grass".

It is a clause, because it contains:

the **subject** **"cows“ and;**

the **predicate** **"eat grass.“**

‡ Example 2 :

"cows eating grass are visible from highway“

This is a complete sentence because;

the **subject** **"cows eating grass“ and;**

the **predicate** **"are visible from the highway“**

Makes complete thought.

- **(b) Predicates & Clause**
- Syntactically a predicate is composed of one or more clauses.
- ‡ The general form of clauses is

  <left-hand-side> :- <right-hand-side>.

- where LHS is a single goal called "*goal*" and RHS is composed of one or more goals, separated by commas, called "*sub-goals*" of the goal on left-hand side.
- The symbol " **:-** " is pronounced as "it is the case" or "such that"

‡ The structure of a clause in logic program

$$\underbrace{\overbrace{\text{pred (functor(var1, var2))}}^{\text{head}} \quad \text{:-} \quad \overbrace{\underbrace{\text{pred(var1)}}_{\text{literal}} \text{ , } \underbrace{\text{pred(var2)}}_{\text{literal}}}^{\text{body}}}_{\text{clause}}$$

Literals represent the possible choices in primitive types the particular language. Some of the choices of types of literals are often integers, floating point, Booleans and character strings.

‡ Example : **grand_parent (X, Z) :- parent(X, Y), parent(Y, Z).**

parent (X, Y) :- mother(X, Y).

parent (X, Y) :- father(X, Y).

Read as if x is mother of y then x is parent of y

‡ Interpretation:

- * A clause specifies the conditional truth of the goal on the LHS; goal on LHS is assumed to be true if the sub-goals on RHS are all true. A predicate is true if at least one of its clauses is true.

- * An **individual "X"** is the **grand-parent of "Z"** if a **parent** of that same **"X"** is **"Y"** and **"Y"** is the **parent** of that **"Z".**

$$\begin{tikzcd}
X \arrow[r, "\text{(X is parent of Y)}"] \arrow[rr, bend right, "\text{(X is grand parent of Z)}"'] & Y \arrow[r, "\text{(Y is parent of Z)}"] & Z
\end{tikzcd}$$

- * An **individual "X"** is a **parent of "Y"** if **"Y" is the mother of "X"**

$$\begin{tikzcd}
X \arrow[r, shift left, "\text{(X is parent of Y)}"] \arrow[r, shift right, "\text{(X is mother of Y)}"'] & Y
\end{tikzcd}$$

- * An **individual "X"** is a **parent of "Y"** if **"Y" is the father of "X"**.

$$\begin{tikzcd}
X \arrow[r, shift left, "\text{(X is parent of Y)}"] \arrow[r, shift right, "\text{(X is father of Y)}"'] & Y
\end{tikzcd}$$

(c) Unit Clause - or Fact

Unlike the previous example of conditional truth, one often encounters unconditional relationships that hold.

‡ In Prolog the clauses that are unconditionally true are called *unit clause* or ***fact*** .

‡ Example :

Unconditionally relationships say

'X' is the father of 'Y'

is unconditionally true.

This relationship as a *Prolog clause* is

father(X, Y) :- true.

Interpreted as relationship of father between **X** and **Y** is always true; or simply stated as **X** is father of **Y** .

‡ Goal true is built-in in Prolog and always holds.

‡ Prolog offers a simpler syntax to express unit clause or fact

father(X, Y)

- i.e. the " **:- true** " part is simply omitted.