Artificial Intelligence

Lecture 15

Symbolic Logic

- ***Prepared by:***

  Md. Mijanur Rahman, Prof. Dr.

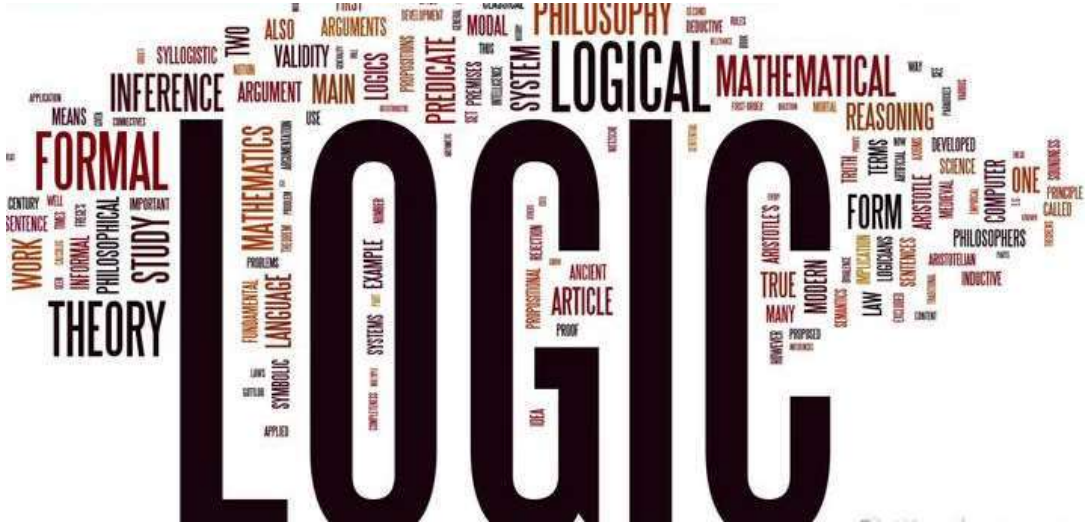
  Dept. of CSE, Jatiya Kabi Kazi Nazrul Islam University

  Email: mijanjkkniu@gmail.com

Symbolic Logic

Lecture Outlines:

- …
- PL: Properties of Statements
- PL: Inference Rules
- Predicate logic
- Predicate Logic: Terminology
- …



PL: Properties of Statements

- **Satisfiable:** A statement is satisfiable if there is some interpretation for which it is true. For example, P is satisfiable.
- **Contradiction**: A sentence/statement is contradiction (unsatisfiable) if there is no interpretation for which it is true. For example, (P&~P).
- **Valid**: A sentence is valid if it is true for every interpretation. Valid sentences are also called tautologies. For example, (PV~P).
- **Equivalence**: Two sentences are equivalence if they have the same truth value under every interpretation. For example, P and ~(~P) are equivalence.
- **Logical consequence**: A sentence is a logical consequence of another if it is satisfied by all interpretations which satisfy the first. For example, P is logical consequence of (P&Q) are equivalence.
  - A sentence is said to be a logical consequence of a set of sentences, for a given language, if and only if, using only logic, the sentence must be true if every sentence in the set is true. It can be expressed using inference rules, for instance:

    All X are Y

    All Y are Z

    Therefore, all X are Z.

Equivalent Logical Expressions

- PV(Q&R) = (PVQ) & (PVR)
- ~(P&Q) = ~PV~Q
- P→Q = ~P V Q
- P⇔Q = (P → Q)&(Q → P)

- **Theorem-1:** The sentence s is a logical consequence of s_1, s_2, …, s_n if and only if $s_1 \& s_2 \& \ldots \& s_n \rightarrow s$ is valid.

- **Theorem-2**: The sentence s is a logical consequence of s_1, s_2, …, s_n if and only if $s_1 \& s_2 \& \ldots \& s_n \& \sim s$ is inconsistent.

PL: Inference Rules

- The inference rules of PL provide the means to perform logical proofs and deductions. Some inference rules:
  - **Modus ponens:** From P and P → Q infer Q.
  - **Chain rule:** From P → Q and Q → R, infer P → R.
  - **Substitution:** If *s* is a valid sentence, s_1 derived from *s* by consistent substitution of propositions in *s*, is also valid.
  - **Simplification:** From P&Q infer P.
  - **Conjunction:** From P and from Q, infer P&Q.
  - **Transposition:** From P → Q, infer ~Q → ~P.

Predicate logic…

- The ***propositional logic, is not powerful*** enough for all types of assertions. For 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 another 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).

Predicate logic

- **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 Logic** deals with predicates, which are propositions containing variables. A predicate is an expression of one or more variables defined on some specific domain. A predicate with variables can be made a proposition by either assigning a value to the variable or by quantifying the variable.
  - The following are some examples of predicates −

    Let E(x, y) denote "x = y"
    Let X(a, b, c) denote "a + b + c = 0"
    Let M(x, y) denote "x is married to y"

Predicate Logic: Terminology…

- **Predicate:** Every complete *sentence* contains two parts: a *subject* and a *predicate*. The *subject* is what (or whom) the sentence is about. The *predicate* tells something about the subject;

  Example:

  - A *sentence "Judy {runs}"*. The subject is *Judy* and the predicate is *runs.* Predicate, always includes verb, tells something about the subject.
  - ***Predicate is a verb phrase template that describes a property of objects, or a relation among objects represented by the variables.***

  - **Predicate logic expressions:** The propositional logic operators combine predicates, like: *If (p(....) && (!q(....) || r (....))).*

    Consider the expression with the respective logic symbols || and *&&:*

    x < y || (y < z && z < x)

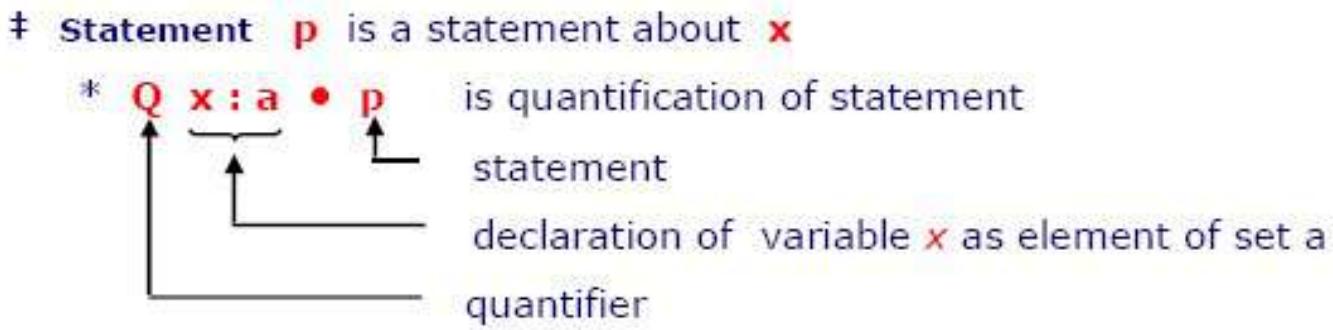
    Assignment for < are 3, 2, 1 for x, y, z and then the value can be *FALSE* or *TRUE;*

    3 < 2 || (2 < 1 && 1 < 3). It is *False.*

Predicate Logic: Terminology…

- **Quantifiers:**
- Generally, a predicate with variables (is called atomic formula) can be made a proposition by applying one of the following two operations to each of its variables :
  - *(1) Assign a value to the variable*; e.g., $x > 1$, if *3* is assigned to x becomes $3 > 1$, and it then becomes a true statement, hence a proposition.
  - *2) Quantify the variable using a quantifier* on formulas of predicate logic (called wff), such as $x > 1$ or $P(x)$, by using Quantifiers on variables.

- **Apply Quantifiers on Variables:**



Predicate Logic: Terminology…

- **Quantifiers:**
  - Quantifiers are two types :
    universal quantifiers , denoted by symbol ∀ and
    existential quantifiers , denoted by symbol ∃

Predicate Logic: Terminology…

- **Universal Quantifier:**
  - **Apply Universal quantifier** ∀ *" For All "*

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

    ‡ Universal quantification: **∀ 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 : ***∀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,
      ∀x means all the objects *x* in the universe
      P(x) is true for every object *x* in the universe

Predicate Logic: Terminology…

- **Universal Quantifier:**

‡ Example : English language to Propositional form

* *"All cars have wheels"*

  $\forall x : car \bullet x \text{ has wheel}$

* $x\ P(x)$

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

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

Predicate Logic: Terminology…

- **Existential Quantifier:**
  - **Apply Existential quantifier** ∃ *" There Exists "*

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

    ‡ Existential quantification: ∃ **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 : ∃ *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
      ∃ *x* means at least one object *x* in the universe
      P(x) is true for least one object *x* in the universe

Predicate Logic: Terminology…

- **Existential Quantifier:**

‡ Example : English language to Propositional form

* " *Someone loves you* "

  $\exists$ *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

Predicate Logic: Terminology…

- **Formula**: A formula is a type of abstract object, a token of which is a symbol or string of symbols which may be interpreted as any meaningful unit in a formal language. It defined recursively as variables, or constants, or functions:
  - Like *f(t1, . . . , tn)*, where *f* is an n-ary function symbol, and *t1, . . . , tn* are terms. Applying predicates to terms produce *atomic formulas*.

- **Atomic formulas:** An atomic formula (or simply atom) is a formula that contains no logical connectives or a formula that has no strict sub-formulas. An atomic formula is one of the form:
  - $t_1 = t_2$, where t_1 and t_2 are terms, or
  - $R(t_1, \ldots, t_n)$, where R is an n-ary relation symbol, and $t_1, \ldots, t_n$ are terms.
  - $\neg a$ is a formula when a is a formula.
  - $(a \wedge b)$ and $(a \vee b)$ are formula when a and b are formula.

Predicate Logic: Terminology…

- **Formulas:**

  – *Atoms* are thus the simplest well-formed formulas of the logic.

  – *Compound formulas* are formed by combining the atomic formulas using the logical connectives.

  – *Well-formed formula* ("wff") is a symbol or string of symbols (a formula) generated by the formal grammar of a formal language.

‡ **Compound formula :** example

$$((((a \wedge b) \wedge c) \vee ((\neg a \wedge b) \wedge c)) \vee ((a \wedge \neg b) \wedge c))$$

Predicate Logic: Terminology…

- **Computable Functions and Predicates:** The objective is to define *class of functions* ***C*** *computable in terms of* ***F***. This is expressed as ***C { F }.*** Example: A conditional expression to define *factorial n (n!)*

◇ The expressions defining n! , n= 5, recursively are :

$n! = n \times (n-1)!$ for $n \geq 1$

$5! = 1 \times 2 \times 3 \times 4 \times 5 = 120$

$0! = 1$

The above definition incorporates an instance that the product of no numbers ie $0! = 1$, then only, the recursive relation $(n + 1)! = n! \times (n+1)$ works for $n = 0$.

◇ Now use conditional expressions

$n! = (n = 0 \rightarrow 1,\ n \neq 0 \rightarrow n . (n - 1) !)$

to define functions recursively.

◇ Example: Evaluate 2! according to above definition.

$2! = (2 = 0 \rightarrow 1,\ 2 \neq 0 \rightarrow 2 . (2 - 1)!)$

$= 2 \times 1!$

$= 2 \times (1 = 0 \rightarrow 1,\ 1 \neq 0 \rightarrow 1 . (1 - 1)!)$

$= 2 \times 1 \times 0!$

$= 2 \times 1 \times (0 = 0 \rightarrow 1,\ 0 \neq 0 \rightarrow 0 . (0 - 1)!)$

$= 2 \times 1 \times 1$

$= 2$

- **Expression:**

" if p_1 then e_1 else if p_2 then e_2 . . . else if p_n then e_n" . i.e., $(p_1 \rightarrow e_1, p_2 \rightarrow e_2, \ldots\ldots p_n \rightarrow e_n)$

Here $p_1, p_2, \ldots . p_n$ are propositional expressions taking the values T or F for true and false respectively. The value of $(p_1 \rightarrow e_1, p_2 \rightarrow e_2, \ldots\ldots .p_n \rightarrow e_n)$ is the value of the *e* corresponding to the first *p* that has value T.

Predicate Logic: Terminology…

- **Production rules:**
  - The other most popular approach to Knowledge representation is to use production rules, sometimes called IF-THEN rules. The remaining two other types of KR are *semantic net* and *frames*. Already discussed earlier classes. Examples of production rules :
    - IF condition THEN action
    - IF premise THEN conclusion
    - IF proposition p_1 and proposition p_2 are true THEN proposition p_3 is true.

- The advantages of production rules :
  - they are modular,
  - each rule define a small and independent piece of knowledge.
  - new rules may be added and old ones deleted
  - rules are usually independently of other rules.
- The production rules as knowledge representation mechanism are used in the design of many *"Rule-based systems*" also called "Production systems" .

Predicate Logic: Terminology

- **Types of Production rules:** Three major types of rules used in the Rule-based production systems:

- **Knowledge Declarative Rules:** These rules state all the facts and relationships about a problem. These rules are a part of the knowledge base.

  IF inflation rate declines

  THEN the price of gold goes down.

- **Inference Procedural Rules:** These rules advise on how to solve a problem, while certain facts are known. These rules are part of the inference engine.

  IF the data needed is not in the system

  THEN request it from the user.

- **Meta rules:** These are rules for making rules. Meta-rules reason about which rules should be considered for firing; i.e., Meta-rules specify which rules should be considered and in which order they should be invoked.

  IF the rules which do not mention the current goal in their premise,

  AND there are rules which do mention the current goal in their premise,

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

Symbolic Logic

To be continued…