First Order Predicate Calculus

- FOPC is a flexible, well-understood, and computationally tractable approach to the representation of knowledge.
- It provides a strong computational basis for the verifiability, inference, and expressiveness requirement.
- The most attractive feature of FOPC:
    - It makes very few specific commitments as to how things ought to be represented.
    - The represented world consists of objects, properties of objects, and relations among objects.

First Order Predicate Calculus

$$
\begin{array}{rcl}
\mathit{Formula} & \rightarrow & \mathit{AtomicFormula} \\
& | & \mathit{Formula}\ \mathit{Connective}\ \mathit{Formula} \\
& | & \mathit{Quantifier}\ \mathit{Variable}, \ldots\ \mathit{Formula} \\
& | & \neg\ \mathit{Formula} \\
& | & (\mathit{Formula}) \\
\\
\mathit{AtomicFormula} & \rightarrow & \mathit{Predicate}(\mathit{Term}, \ldots) \\
\\
\mathit{Term} & \rightarrow & \mathit{Function}(\mathit{Term}, \ldots) \\
& | & \mathit{Constant} \\
& | & \mathit{Variable} \\
\\
\mathit{Connective} & \rightarrow & \wedge \mid \vee \mid \Rightarrow \\
\mathit{Quantifier} & \rightarrow & \forall \mid \exists \\
\mathit{Constant} & \rightarrow & A \mid \mathit{VegetarianFood} \mid \mathit{Maharani} \cdots \\
\mathit{Variable} & \rightarrow & x \mid y \mid \cdots \\
\mathit{Predicate} & \rightarrow & \mathit{Serves} \mid \mathit{Near} \mid \cdots \\
\mathit{Function} & \rightarrow & \mathit{LocationOf} \mid \mathit{CuisineOf} \mid \cdots
\end{array}
$$

Elements of FOPC

- **Term –** the FOPC device for representing objects.
- Three basic building blocks: constants, functions, variables
    - **Constants**
        - Specific objects in the world being described.
        - Depicted as single capitalized letters *A*, *B*, or proper nouns such as *Maharani*, *Ram*
    - **Functions**
        - Concepts that are expressed in English as genitives,
            - *the location of Maharani*, or *Maharani's location*
            - *LocationOf* (*Maharani*)
        - Refer to unique objects, though appearing similarly as predicates

Elements of FOPC

- Three basic building blocks: constants, functions, variables
  - **Variables**
    - Depicted as single lower-case letters
    - Ability to make assertions and draw inferences about objects without having to make reference to any particular named object

Elements of FOPC

- **Predicates –** Relations that hold among objects.
  - Predicates are symbols refer to, or name, the relations that hold among some fixed number of objects in a given domain
    - *Serves*(*Maharani*,*VegetarianFood*) – a two-place predicate
    - *Restaurant*(*Maharani*) – a one-place predicate
  - Complex formula, through the use of **logical connectives**
  - (14.17) *I only have five dollars and I don't have a lot of time.*
    - $Have(Speaker, FiveDollars) \wedge \neg Have(Speaker, LotOfTime)$

The Semantics of FOPC

- How various objects, properties, and relations presented on a FOPC acquire their meanings?

  by virtue of their correspondence to objects, properties, and relations out in the external world being modeled by the knowledge base
- FOPC sentences can therefore be assigned a value of *True* or *False*
- The interpretations of formulas involving logical connectives is based on the meaning of the components in the formulas combined with the meaning of connectives they contain.

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$
False	*False*	*True*	*False*	*False*	*True*
False	*True*	*True*	*False*	*True*	*True*
True	*False*	*False*	*False*	*True*	*False*
True	*True*	*False*	*True*	*True*	*True*

Variables and Quantifiers

- Variables are used in two ways in FOPC:
    - To refer to particular anonymous objects and
    - To refer generically to all objects in a collection

– The two uses are made possible through the use of **quantifiers**.

- The two operators are the existential quantifier - $\exists$ (“*there exists*”) and the universal quantifier - $\forall$ (“*for all*”)
- The need for existential quantifier is due to the presence of an indefinite noun phrase

(14.19) *a restaurant that serves Mexican food near ICSI*

$$\exists x\ Restaurant(x) \land Serves(x, MexicanFood) \land Near(LocationOf(x), LocationOf(ICSI))$$

Variables and Quantifiers

- For this sentence to be true there must be at least one object such that if substituted for x, the resulting sentence would be true
- If *Gateway* is a Mexican restaurant near ICSI, then:
  - Substituting for x results in:

$$
\begin{aligned}
&Restaurant(Gateway)\\
&\qquad \wedge\ Serves\ (Gateway, MexicanFood)\\
&\qquad \wedge\ Near\ (LocationOf(Gateway), LocationOf(ICSI))
\end{aligned}
$$

  - The sentence will be *true* if all of its three atomic formulas are *true*.

Variables and Quantifiers

- $\forall$ operator states that for the formula to be true, the substitution of *any* object in the knowledge base for the universally quantified variable should result in a true formula.
- Consider the following example:
  - (14.20) All vegetarian restaurants serve vegetarian food.

    $$\forall x\ VegetarianRestaurant(x) \Rightarrow Serves(x, VegetarianFood)$$

  - Case 1: Set of objects consisting of vegetarian restaurants:

    $$VegetarianRestaurant(Maharani) \Rightarrow Serves(Maharani, VegetarianFood)$$

  - If consequent is true or both antecedent and the consequent have the value *True*, then the sentence itself is *True*.

Variables and Quantifiers

- Consider the following example:
  - (14.20) All vegetarian restaurants serve vegetarian food.

    $$\forall x\ VegetarianRestaurant(x) \Rightarrow Serves(x, VegetarianFood)$$

  - Case 2: Set of a objects that are not vegetarian restaurants:

    $$VegetarianRestaurant(Gateway) \Rightarrow Serves(Gateway, VegetarianFood)$$

  - Since the antecedent of the implication is *False*, the sentence is always *True* satisfying the $\forall$ constraint.
  - There is no restrictions on objects that can be substituted for x by this kind of reasoning.

Inference

- Inference
  - The ability to add valid new propositions to a knowledge base, or
  - To determine the truth of propositions not explicitly contained within a knowledge base.

– **Modus ponens –** inference method provided by FOPC.

– If the left-hand side of an implication rule is present in the knowledge base, then the right-hand side of the rule can be inferred.

$$\frac{\begin{array}{c}\alpha \\ \alpha \Rightarrow \beta\end{array}}{\beta} \qquad \frac{\begin{array}{l}\mathit{VegetarianRestaurant}(\mathit{Rudys}) \\ \forall x\ \mathit{VegetarianRestaurant}(x) \Rightarrow \mathit{Serves}(x, \mathit{VegetarianFood})\end{array}}{\mathit{Serves}(\mathit{Rudys}, \mathit{VegetarianFood})}$$

– The formula *VegetarianRestaurant*(*Rudys*) matches the antecedent thus using modus ponen concludes *Serves*(*Rudys*, *VegetarianFood*)

Inference

- Modus ponens used in two ways:
- Forward chaining:
  - As soon as new fact is added to **kb** all applicable implication rules are found and applied, each resulting in addition of new facts to the **kb**.
  - All inference is preformed in advance, hence facts will be present always.
- Backward chaining:
  - Modus ponen run in reverse.
    - Check if the query formula is present in the kb
    - If not, search for applicable implication rule (consequent matches the query) present in kb
    - Query is proved if any of the antecedent is shown to be true

Inference

- Backward chaining:
  - **Prolog** programming language is a backward chaining system.

$VegetarianRestaurant(Rudys)$

$\forall x\ VegetarianRestaurant(x) \Rightarrow Serves(x, VegetarianFood)$

? $Serves(Rudys, VegetarianFood)$.

True.

after substituting the constant (*Rudys*) for variable *(x)*, prove the antecedent of the rule