2.2 The Predicate Calculus:

In propositional calculus, each atomic symbol (**P, Q, etc**.) denotes a proposition of some complexity. There is no way to access the components of an individual assertion. Predicate calculus provides this ability. For example, instead of letting a single propositional symbol, **P**, denote the entire sentence "**it rained on Tuesday**," we can create a predicate weather that describes a relationship between a date and the weather: **weather(tuesday, rain)**. Through inference rules we can manipulate predicate calculus expressions, accessing their individual components and inferring new sentences.

Predicate calculus also allows expressions to contain variables. Variables let us create general assertions about classes of entities. For example, we could state that for all values of **X**, where **X** is a day of the week, the statement **weather(X, rain)** is true; i.e., it rains every day. As with propositional calculus, we will first define the syntax of the language and then discuss its semantics.

2.2.1 The Syntax of Predicates and Sentences:

Before defining the syntax of correct expressions in the predicate calculus, we define an alphabet and grammar for creating the *symbols* of the language. This corresponds to the lexical aspect of a programming language definition. Predicate calculus symbols, like the ***tokens*** in a programming language, are irreducible syntactic elements: they cannot be broken into their component parts by the operations of the language.

DEFINITIONS: PREDICATE CALCULUS SYMBOLS

The alphabet that makes up the symbols of the predicate calculus consists of:

1. The set of letters, both **uppercase A..Z** and **lowercase a..z**, of the English alphabet.
2. The set of digits, **0, 1, ... ,9**.
3. The underscore "_".

Symbols in the predicate calculus begin with a letter and are followed by any sequence of these legal characters.

Examples: of characters **not in the alphabet** include: **# % @ / & ""**

Examples: **of Legal predicate** calculus symbols include:

George, fire3, tom_and_bill, XXXX, friends, of, …

Examples: of strings that are **not legal symbols** are:

3jack, "no blanks allowed", ab%cd, ***71, duck!!!

Symbols are used to denote ***objects***, ***properties***, or ***relations*** in world of discourse. As with most programming languages, the use of "words" that suggest the symbol's intended meaning assists us in understanding program code. Thus, even though **l(g, k)** and **likes(george, kate)** are formally equivalent (i.e., they have the same structure), the second can be of great help (for human readers) in indicating what relationship the expression represents. It must be stressed that these descriptive names are intended solely to improve the readability of expressions. The only meaning that predicate calculus expressions may be said to have is through their formal semantics. ***Parentheses*** "**( )**", ***commas*** "**,**", and ***periods*** "**.**" are used solely to construct well-formed expressions and do not denote objects or relations in the world. These are called ***improper symbols.***

Predicate calculus symbols may represent ***variables, constants, functions,*** or ***predicates***. **Constants** name specific objects or properties in the world. Constant symbols must begin with a lowercase letter. Thus **george, tree, tall,** and **blue** are examples of well-formed constant symbols. The constants **true** and **false** are reserved as ***truth symbols.*** **Variable** symbols are used to designate general classes of objects or properties in the world Variables are represented by symbols beginning with an uppercase letter. Thus **George, BILL, and KAte** are legal variables, whereas **geORGE** and **bill** are not.

Predicate calculus also allows functions on objects in the world of discourse. Function symbols (like constants) begin with a lowercase letter. Functions denote a mapping of one or more elements in a set (called the **domain** of the function) into a unique element of another set (the **range** of the function). Elements of the domain and range are objects in the world of discourse. In addition to common arithmetic functions such as addition and multiplication,

functions may define mappings between nonnumeric domains. Note that our definition of predicate calculus symbols does not include numbers or arithmetic operators. The number system is not included in the predicate calculus primitives; instead it is defined axiomatically using "pure" predicate calculus as a basis. While the particulars of this derivation are of theoretical interest, they are less important to the use of predicate calculus as an AI representation language. For convenience, we assume this derivation and include arithmetic in the language. Every function symbol has an associated **arity**, indicating the number of elements in the domain mapped onto each element of the range. Thus father could denote a function of **arity 1** that maps people onto their (unique) male parent, plus could be a function of **arity 2** that maps two numbers onto their arithmetic sum. *A **function expression*** is a function symbol followed by its arguments. The arguments are elements from the domain of the function; the number of arguments is equal to the arity of the function. The arguments are enclosed in parentheses and separated by commas. For example: **f(X,Y), father(john), price(bananas)** are all well-formed function expressions.

Each function expression denotes the mapping of the arguments onto a single object in the range, called the ***value*** of the function. For example, if father is a unary function, then : **father(david)** is a function expression whose value is **george** (i.e. ***george father of david***). If plus is a function of arity 2, with domain the integers, then: **plus(2,3)** is a function expression whose value is the integer 5. The act of replacing a function with its value is called ***evaluation***. The concept of a predicate calculus symbol or term is formalized in the following definition:

DEFINITION: SYMBOLS and TERMS

Predicate calculus symbols include:

1. ***Truth symbols*** **true** and **false** (these are reserved symbols).
2. ***Constant symbols*** are symbol expressions having the first character **lowercase**.
3. ***Variable symbols*** are symbol expressions beginning with an **uppercase** character.
4. ***Function symbols*** are symbol expressions having the first character **lowercase**.

Functions have an attached arity indicating the number of elements of the domain

mapped onto each element of the range.

A ***function expression*** consists of a function constant of arity n, followed by n terms, $\mathbf{t_1, t_2 , \dots , t_n}$, enclosed in parentheses and separated by commas.

A predicate calculus ***term*** is either a constant, variable, or function expression. Thus, a predicate calculus ***term*** may be used to denote objects and properties in a problem domain. Examples of terms are: **cat, times(2,3), X, blue, mother(jane).**

Symbols in predicate calculus may also represent predicates. Predicate symbols, like constants and function names, begin with a lowercase letter. A predicate names a relationship between zero or more objects in the world. The number of objects so related is the arity of the predicate, Examples of predicates are : **likes, equals, on.**

Atomic sentence: The most primitive unit of the predicate calculus language, atomic sentence is a predicate constant of arity **n** followed by **n** terms enclosed in parentheses and separated by commas.

Examples of **atomic sentences** are:

likes(george, kate)	**likes(X, george)**
likes(george, susie)	**likes(X,X)**
likes(george, sarah, tuesday)	**friends(bill, richard)**
friends(bill, george)	**friends(father_of(david), father_of(andrew))**
helps(bill, george)	**helps(richard, bill)**

The predicate symbols in these expressions are likes, friends, and helps. A predicate symbol may be used with different numbers of arguments. In this example there are two different likes, one with two and the other with three arguments. When a predicate symbol is used in sentences with different arities, it is considered to represent two different relations. Thus, a predicate relation is defined by its name and its arity. There is no reason that the two different likes cannot make up part of the same description of the world however. this is avoided because it can often cause confusion. In the predicates above, **bill, george, kate, etc**., are constant symbols and represent objects in the problem domain. The

arguments to a predicate are terms and may also include variables or function expressions. For example:

friends(father_of(david), father_of(andrew))

is a predicate describing a relationship between two objects in a domain of discourse. These arguments are represented as function expressions whose mappings (given that the **father_of david** is **george** and the **father_of andrew is allen**) form the parameters of the predicate. If the function expressions are evaluated, the expression becomes: **friends(george, allen)**

DEFINITION: PREDICATES and ATOMIC SENTENCES

Predicates have an associated positive integer referred to as the arity or "argument number" for the predicate. Predicates with the same name but different arities are considered distinct. An atomic sentence is a predicate constant of arity n, followed by n terms, $t_1, t_2, \ldots, t_n$, enclosed in parentheses and separated by commas. The truth values, true and false, are also atomic sentences.

Atomic sentences are also called ***atomic expressions, atoms,*** or ***propositions.***

We may combine atomic sentences using logical operators to form *sentences* in the predicate calculus. These are the same logical connectives used in propositional calculus: $\wedge$, $\vee$, $\neg$, $\rightarrow$, and $\equiv$

When a variable appears as an argument in a sentence, it refers to unspecified objects in the domain. First order predicate calculus includes two symbols, the *variable quantifiers* $\forall$ and $\exists$, that constrain the meaning of a sentence containing a variable. A quantifier is followed by a variable and a sentence, such as :

$\exists$Y friends(Y, peter)

$\forall$X likes(X, ice_cream)

The ***universal quantifier,*** $\forall$, indicates that the sentence is true for all values of the variable.

In the example, ∀**X likes(X, ice_cream)** is true for all values in the domain of the definition of **X**. The ***existential quantifier,*** ∃, indicates that the sentence is true for at least one value in the domain. ∃**Y friends(Y, peter)** is true if there is at least one object, indicated by **Y** that is a friend of peter.

DEFINITION: PREDICATE CALCULUS SENTENCES

1. Every **atomic sentence** is a sentence.
2. If **S** is a sentence, then so is its negation, ¬**S**
3. If S_1 and S_2 are sentences, then so is their conjunction, $S_1 \wedge S_2$
4. If S_1 and S_2 are sentences, then so is their disjunction, $S_1 \vee S_2$
5. If S_1 and S_2 are sentences, then so is their implication, $S_1 \rightarrow S_2$,
6. If S_1 and S_2 are sentences, then so is their equivalence, $S_1 \equiv S_2$.
7. If **X** is a variable and **S** a sentence, then ∀**X** is a sentence.
8. If **X** is a variable and **S** a sentence, then ∃**X** s is a sentence.

Examples: Let **times** and **plus** be function symbols of arity 2, and let **equal** and **foo** be predicate symbols with arity 2 and 3, respectively.

plus(two, three) ***is a function and thus not an atomic sentence.***

equal(plus(two, three), five) ***is an atomic sentence***.

equal(plus(2, 3), seven) ***is an atomic sentence***.

Note that this sentence, given the standard interpretation of **plus** and **equal** is false. Well-formalness and truth value are independent issues.

∃X foo(X, two, plus(two, three)) ∧ equal(plus(two, three), five)

is a sentence because both conjuncts are sentences.

(foo(two, two, plus(two, three))) →(equal(plus(three, two), five) ≡ true)

is a sentence because all its components are sentences, appropriately connected by logical operators.