LECTURE 18
Chapter 13
Part 1: Predicate Languages
Predicate Languages are also called First Order Languages.

The same applies to the use of terms for Propositional and Predicate Logic. Propositional and Predicate Logics are called Zero Order and First Order Logics, and we will use both terms equally.

We usually work with different predicate languages, depending on what applications we have in mind. All of those predicate languages have some common features, and we begin with these.
Predicate Languages Components

Propositional Connectives
Predicate Languages extend a notion of the propositional languages so we define the set $CON$ of their propositional connectives as follows:

The set $CON$ of propositional connectives is a finite and non-empty and

$$CON = C_1 \cup C_2$$

where $C_1, C_2$ are the sets of one and two arguments connectives, respectively.

Parenthesis
As in the propositional case, we adopt the signs ( and ) for our parenthesis., i.e. we define a set $PAR$ as

$$PAR = \{(,\, )\}$$
Predicate Languages Components

Quantifiers
We adopt two quantifiers; the universal quantifier denoted by $\forall$ and the existential quantifier denoted by $\exists$, i.e. we have the following set $Q$ of quantifiers

$$Q = \{\forall, \exists\}$$

In a case of the classical logic and the logics that extend it, it is possible to adopt only one quantifier and to define the other in terms of it and propositional connectives.

Such definability is impossible in a case of some non-classical logics, for example the intuitionistic logic.

But even in the case of classical logic the two quantifiers express better the common intuition, so we adopt the two of them.
Predicate Languages Components

Variables
We assume that we always have a countably infinite set $\text{VAR}$ of variables, i.e. we assume that $\text{cardVAR} = \aleph_0$

We denote variables by $x, y, z, ...$, with indices, if necessary. we often express it by writing

$\text{VAR} = \{x_1, x_2, ...\}$
Predicate Languages Components

The set \( CON \) of **propositional connectives** defines a propositional part of the **predicate logic language**. Observe that what really differ one **predicate language** from the other is the choice of additional symbols added to the symbols just described.

These **additional symbols** are: predicate symbols, function symbols, and constant symbols.

A **particular** predicate language is **determined** by specifying these **additional sets of symbols**. They are defined as follows.
Predicate Languages Components

Predicate symbols
Predicate symbols represent relations
Any predicate language must have at least one predicate symbol
Hence we assume that any predicate language contains a non empty, finite or countably infinite set

$$\mathbf{P}$$

of predicate symbols, i.e. we assume that

$$0 < \text{card} \mathbf{P} \leq \aleph_0$$

We denote predicate symbols by $P, Q, R, ...$, with indices, if necessary
Each predicate symbol $P \in \mathbf{P}$ has a positive integer $\#P$ assigned to it; when $\#P = n$ we call $P$ an $n$-ary (n-place) predicate (relation) symbol
Predicate Languages Components

—bf Function symbols

We assume that any predicate language contains a finite (may be empty) or countably infinite set $F$ of function symbols

I.e. we assume that

$$0 \leq \text{card } F \leq \aleph_0$$

When the set $F$ is empty we say that we deal with a language without functional symbols

We denote functional symbols by $f, g, h, ...$ with indices, if necessary

Similarly, as in the case of predicate symbols, each function symbol $f \in F$ has a positive integer $\# f$ assigned to it; if $\# f = n$ then $f$ is called an n-ary (n-place) function symbol
Predicate Languages Components

Constant symbols

We also assume that we have a finite (may be empty) or countably infinite set of constant symbols. I.e. we assume that

\[ 0 \leq \text{card} C \leq \aleph_0 \]

The elements of \( C \) are denoted by \( c, d, e \ldots \), with indices, if necessary. We often express it by putting

\[ C = \{ c_1, c_2, \ldots \} \]

When the set \( C \) is empty we say that we deal with a language without constant symbols.
Alphabet of Predicate Languages

Sometimes the **constant symbols** are defined as **0-ary function symbols**, i.e. we have that

\[ C \subseteq F \]

We single them out as a separate set for our convenience.

We assume that all of the above sets of symbols are **disjoint**.

**Alphabet**

The union of all of above disjoint sets of symbols is called the **alphabet** \( \mathcal{A} \) of the **predicate language**, i.e. we define

\[
\mathcal{A} = \text{VAR} \cup \text{CON} \cup \text{PAR} \cup \text{Q} \cup \text{P} \cup \text{F} \cup \text{C}
\]
Observe, that once the set of propositional connectives is fixed, the predicate language is determined by the sets $P$, $F$ and $C$

We use the notation

$$\mathcal{L}(P, F, C)$$

for the predicate language $\mathcal{L}$ determined by $P$, $F$, $C$

If there is no danger of confusion, we may abbreviate $\mathcal{L}(P, F, C)$ to just $\mathcal{L}$

If the set of propositional connectives involved is not fixed, we also use the notation

$$\mathcal{L}_{\text{CON}}(P, F, C)$$

to denote the predicate language $\mathcal{L}$ determined by $P$, $F$, $C$ and the set of propositional connectives $\text{CON}$
Predicate Languages Notation

We sometimes allow the same symbol to be used as an n-place relation symbol, and also as an m-place one; no confusion should arise because the different uses can be told apart easily.

**Example**

If we write $P(x, y)$, the symbol $P$ denotes **2-argument** predicate symbol.

If we write $P(x, y, z)$, the symbol $P$ denotes **3-argument** predicate symbol.

Similarly for **function symbols**.
Two more Predicate Language Components

Having defined the alphabet we now complete the formal definition of the predicate language by defining two more components:

the set $T$ of all terms and

the set $F$ of all well formed formulas

of the language $\mathcal{L}(P, F, C)$
Set of Terms

Terms

The set $T$ of terms of the predicate language $\mathcal{L}(P, F, C)$ is the smallest set

$$T \subseteq \mathcal{A}^*$$

meeting the conditions:

1. any variable is a term, i.e. $\text{VAR} \subseteq T$
2. any constant symbol is a term, i.e. $C \subseteq T$
3. if $f$ is an $n$-place function symbol, i.e. $f \in F$ and $\#f = n$ and $t_1, t_2, ..., t_n \in T$, then $f(t_1, t_2, ..., t_n) \in T$
Terms Examples

Example 1
Let $f \in F$, $\#f = 1$, i.e. $f$ is a 1-place function symbol
Let $x, y$ be variables, $c, d$ be constants, i.e. $x, y \in VAR$, $c, d \in C$
Then the following expressions are terms:

$$x, y, f(x), f(y), f(c), f(d), ff(x), ff(y), ff(c), ff(d), ...$$

Example 2
Let $F = \emptyset$, $C = \emptyset$
In this case terms consists of variables only, i.e.

$$T = VAR = \{x_1, x_2, ...\}$$
Terms Examples

Directly from the Example 2 we get the following

REMARK

For any predicate language $\mathcal{L}(P, F, C)$, the set $T$ of its terms is always non-empty

Example 3

Let $f \in F$, $\#f = 1$, $g \in F$, $\#g = 2$, $x, y \in VAR, c, d \in C$

Some of the terms are the following:

$$f(g(x, y)), f(g(c, x)), g(ff(c), g(x, y)),$$

$$g(c, g(x, f(c))), g(f(g(x, y)), g(x, f(c))) \ldots$$
Terms Notation

From time to time, the logicians are and we may be informal about how we write terms.

Example
If we denote a 2-place function symbol $g$ by $+$, we may write $x + y$ instead of $+(x, y)$.

Because in this case we can think of $x + y$ as an unofficial way of designating the "real" term $+(x, y)$.
Atomic Formulas

Before we define the set of formulas, we need to define one more set; the set of atomic, or elementary formulas.

Atomic formulas are the simplest formulas as the propositional variables were in the case of propositional languages.
Atomic Formulas

Definition
An atomic formula of a predicate language $\mathcal{L}(P, F, C)$ is any element of $\mathcal{A}^*$ of the form

$$R(t_1, t_2, \ldots, t_n)$$

where $R \in P$, $\#R = n$ and $t_1, t_2, \ldots, t_n \in T$

I.e. $R$ is n-ary relational symbol and $t_1, t_2, \ldots, t_n$ are any terms

The set of all atomic formulas is denoted by $\mathcal{AF}$ and is defined as

$$\mathcal{AF} = \{R(t_1, t_2, \ldots, t_n) \in \mathcal{A}^* : R \in P, t_1, t_2, \ldots, t_n \in T, n \geq 1\}$$
Atomic Formulas Examples

Example 1
Consider a language $\mathcal{L}(\emptyset, \{P\}, \emptyset)$, for $\#P = 1$
Our language

$$\mathcal{L} = \mathcal{L}(\emptyset, \{P\}, \emptyset)$$

is a language without neither functional, nor constant symbols, and with one, 1-place predicate symbol $P$
The set of atomic formulas contains all formulas of the form $P(x)$, for $x$ any variable, i.e.

$$\mathcal{AF} = \{P(x) : x \in \text{VAR}\}$$
Example 2
Let now consider a **predicate language**

\[ \mathcal{L} = \mathcal{L}(\{f, g\}, \{R\}, \{c, d\}) \]

for \( \#f = 1, \#g = 2, \#R = 2 \)

The language \( \mathcal{L} \) has two functional symbols: 1-place symbol \( f \) and 2-place symbol \( g \), one 1-place predicate symbol \( R \), and two constants: \( c, d \)

Some of the atomic formulas in this case are the following.

\[ R(c, d), \ R(x, f(c)), \ R((g(x, y)), f(g(c, x))), \]

\[ R(y, g(c, g(x, f(d)))) \ldots \]
Set of Formulas Definition

Now we are ready to define the set $\mathcal{F}$ of all well formed formulas of any predicate language $\mathcal{L}(P, F, C)$

Definition

The set $\mathcal{F}$ of all well formed formulas, called shortly set of formulas, of the language $\mathcal{L}(P, F, C)$ is the smallest set meeting the following four conditions:

1. Any atomic formula of $\mathcal{L}(P, F, C)$ is a formula, i.e. $\mathcal{AF} \subseteq \mathcal{F}$

2. If $A$ is a formula of $\mathcal{L}(P, F, C)$, $\triangledown$ is an one argument propositional connective, then $\triangledown A$ is a formula of $\mathcal{L}(P, F, C)$, i.e. the following recursive condition holds

   \[
   \text{if } A \in \mathcal{F}, \triangledown \in C_1 \text{ then } \triangledown A \in \mathcal{F}
   \]
Set of Formulas Definition

3. If $A, B$ are formulas of $\mathcal{L}(P, F, C)$ and $\circ$ is a two argument propositional connective, then $(A \circ B)$ is a formula of $\mathcal{L}(P, F, C)$, i.e. the following recursive condition holds

$$\text{If } A \in \mathcal{F}, \nabla \in C_2, \text{ then } (A \circ B) \in \mathcal{F}$$

4. If $A$ is a formula of $\mathcal{L}(P, F, C)$ and $x$ is a variable, then $\forall_x A, \exists_x A$ are formulas of $\mathcal{L}(P, F, C)$, i.e. the following recursive condition holds

$$\text{If } A \in \mathcal{F}, x \in \text{VAR}, \forall, \exists \in Q, \text{ then } \forall x A, \exists x A \in \mathcal{F}$$
Scope of Quantifiers

Another important notion of the predicate language is the notion of a scope of the quantifier. It is defined as follows:

**Definition**

Given formulas $\forall x A$, $\exists x A$, the formula $A$ is said to be in the scope of the quantifier $\forall$, $\exists$, respectively.

**Example 3**

Let $\mathcal{L}$ be a language of the previous Example 2 with the set of connectives $\{\cap, \cup, \Rightarrow, \neg\}$, i.e. let's consider

$$\mathcal{L} = \mathcal{L}_{\{\cap, \cup, \Rightarrow, \neg\}}(\{f, g\}, \{R\}, \{c, d\})$$

for $#f = 1$, $#g = 2$, $#R = 2$

Some of the formulas of $\mathcal{L}$ are the following.

$$R(c, d), \exists y R(y, f(c)), \neg R(x, y),$$

$$(\exists x R(x, f(c)) \Rightarrow \neg R(x, y))$$

$$(R(c, d) \cap \forall z R(z, f(c)))$$

$$\forall y R(y, g(c, g(x, f(c))))$$

$$\forall y \neg \exists x R(x, y)$$
Scope of Quantifiers

The formula $R(x, f(c))$ is in a **scope of the quantifier** $\exists x$ in the formula

$\exists x R(x, f(c))$

The formula $(\exists x R(x, f(c)) \Rightarrow \neg R(x, y))$ is **not in a scope of any quantifier**

The formula $(\exists x R(x, f(c)) \Rightarrow \neg R(x, y))$ is in the **scope** of $\forall y$ in the formula

$\forall y (\exists x R(x, f(c)) \Rightarrow \neg R(x, y))$
Predicate Language Definition

Now we are ready to define formally a **predicate language**. Let $A, T, F$ be the **alphabet**, the set of **terms** and the set of **formulas** as already defined.

**Definition**

A **predicate language** $L$ is a triple

$$L = (A, T, F)$$

As we have said before, the language $L$ is determined by the choice of the symbols of its **alphabet**, namely of the choice of connectives, predicates, functions, and constant symbols.

If we want specifically mention these **choices**, we write

$$L = L_{CON}(P, F, C)$$

or

$$L = L(P, F, C)$$