CHAPTER 13

PREDICATE LANGUAGES

1 Predicate Languages

Propositional Languages are also called Zero Order Languages, as opposed to Predicate Languages that are called First Order Languages. The same applies to the use of terms Propositional and Predicate Logic; they are often called zero Order and First Order Logics and we will use both terms equally.

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

Propositional connectives We define the set of propositional connectives

$$CON$$

in the same way as in the case of the propositional languages. It means that we assume the following.

1. The set of connectives is non-empty and finite, i.e.

$$0 < cardCON < \aleph_0.$$

2. We consider only the connectives with one or two arguments.

Quantifiers We adopt two quantifiers; $\forall$ (for all, the universal quantifier) and $\exists$ (there exists, the existential quantifier), i.e. we have the following set of quantifiers

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

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. It is impossible in a case of some non-classical logics, for example the intuitionistic logic. But even in the case of classical logic two quantifiers express better the common intuition, so we assume that we have two of them.

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

$$PAR = \{(,)\}.$$

Variables We assume that we always have a countably infinite set VAR of variables, i.e. we assume that

$$cardVAR = \aleph_0.$$

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

$$VAR = \{x_1, x_2, ...\}.$$

The set of propositional connectives CON defines a propositional part of the predicate logic language. What really differ one predicate language from the other is the choice of additional symbols to the symbols described above. These are called predicate symbols, function symbols, and constant symbols. I.e. a particular predicate language is determined by specifying the following sets of symbols.

Predicate symbols Predicate symbols represent relations. We assume that we have an non empty, finite or countably infinite set

$$\mathbf{P}$$

of predicate, or relation symbols. I.e. we assume that

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

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; if $\#P = n$ then say P is called an n-ary (n - place) predicate (relation) symbol.

Function symbols We assume that we have a finite (may be empty) or countably infinite set

$$\mathbf{F}$$

of function symbols. I.e. we assume that

$$0 \leq card\mathbf{F} \leq \aleph_0.$$

When the set $\mathbf{F}$ 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 $f \in \mathbf{F}$ has a positive integer $\#f$ assigned to it; if $\#f = n$ then say f is called an n-ary (n - place) function symbol.

Constant symbols We also assume that we have a finite (may be empty) or countably infinite set

$$\mathbf{C}$$

of constant symbols. I.e. we assume that

$$0 \leq card\mathbf{C} \leq \aleph_0.$$

The elements of $\mathbf{C}$ are denoted by $c, d, e...$, with indices, if necessary, what we often express by writing

$$\mathbf{C} = \{c_1, c_2, ...\}.$$

When the set $\mathbf{C}$ is empty we say that we deal with a language without constant symbols.

Sometimes the constant symbols are defined as 0-ary function symbols, i.e. $\mathbf{C} \subset \mathbf{F}$. We single them out as a separate set for our convenience.

Disjoint sets We assume that all of the above sets are disjoint.

Alphabet The union of all of above disjoint sets is called the *alphabet* $\mathcal{A}$ of the predicate language, i.e.

$$\mathcal{A} = VAR \cup CON \cup PAR \cup \mathbf{Q} \cup \mathbf{P} \cup \mathbf{F} \cup \mathbf{C}.$$

Observe, that once the set of propositional connectives is fixed, the predicate language is determined by the sets **P, F** and **C**, so we use the notation

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

for the predicate language $\mathcal{L}$ determined by **P, F** and **C**. If there is no danger of confusion, we may abbreviate $\mathcal{L}(\mathbf{P}, \mathbf{F}, \mathbf{C})$ to just $\mathcal{L}$. If for some reason we need to stress the set of propositional connectives involved, we will also use the notation

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

to denote the predicate language $\mathcal{L}$ determined by **P, F, C** and the set of propositional connectives CON.

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. Similarly for function symbols.

Having defined the basic elements of syntax, the alphabet, we can now complete the formal definition of the predicate language by defining two more complex sets: the set T of all terms and the set $\mathcal{F}$ of all well formed formulas of the language $\mathcal{L}(\mathbf{P}, \mathbf{F}, \mathbf{C})$.

Terms The set
$$T$$
of terms of the predicate language $\mathcal{L}(\mathbf{P},\mathbf{F},\mathbf{C})$ is the smallest set $T \subset \mathcal{A}^*$ meeting the conditions:

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

Example If $f \in \mathbf{F}, \#f = 1$, i.e. f is a one place function symbol, x, y are variables, c, d are constants, i.e. $x, y \in VAR, c, d \in \mathbf{C}$, then the following are terms:

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

Example If $\mathbf{F} = \emptyset, \mathbf{C} = \emptyset$, then the set T of terms consists of variables only, i.e.
$$T = VAR = \{x_1, x_2, ...\}.$$
From the above we get the following observation.

Remark 1.1 *For any predicate language $\mathcal{L}(\mathbf{P},\mathbf{F},\mathbf{C})$, the set T of its terms is always non-empty.*

Example If $f \in \mathbf{F}, \#f = 1$, $g \in \mathbf{F}, \#g = 2$, $x, y \in VAR, c, d \in \mathbf{C}$, then some of the terms are the following:

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

From time to time, the logicians are and we may be informal about how we write terms. For instance, if we denote a two place function symbol g by $+$, we may write $x + y$ instead $+(x, y)$. Because in this case we can think of $x + y$ as an unofficial way of designating the "real" term $+(x, y)$, or even $g(x, y)$.

Before we define the set of formulas, we need to define one more set; the set of atomic, or elementary formulas. They are the "smallest" formulas as were the propositional variables in the case of propositional languages.

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

$$R(t_1, t_2, ..., t_n),$$

where $R \in \mathbf{P}, \#R = n$, i.e. R is n-ary relational symbol and $t_1, t_2, ..., t_n$ are terms. The set of all atomic formulas is denoted by $\mathcal{AF}$ and is defines as

$$\mathcal{AF} = \{R(t_1, t_2, ..., t_n) \in \mathcal{A}^* : R \in \mathbf{P},\ t_1, t_2, ..., t_n \in T,\ \#R = n,\ n \geq 1\}.$$

Example Consider a language

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

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

$$\mathcal{AF} = \{P(x) : x \in VAR\}.$$

Example Let now

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

for #f = 1, #g = 2 , #R = 2, i.e. $\mathcal{L}$ has two functional symbols: one -place symbol f and two-place symbol g; one two-place predicate symbol R, and two constants: c,d. Some of the atomic formulas in this case are the following.

$$R(c, d),\ R(x, f(c)),\ R(f(g(x, y)), f(g(c, x))),\ R(y, g(c, g(x, f(c)))).$$

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

Formulas The set

$$\mathcal{F}$$

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

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

$$\mathcal{AF} \subseteq \mathcal{F};$$

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

$$if\ A \in \mathcal{F}, \bigtriangledown \in C_1,\ then\ \bigtriangledown A \in \mathcal{F};$$

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

$$if\ A \in \mathcal{F}, \bigtriangledown \in C_2,\ then\ (A \circ B) \in \mathcal{F};$$

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

$$if\ \ A \in \mathcal{F},\ \ x \in VAR,\ \ \forall, \exists \in \mathbf{Q}\ \ then\ \forall x A,\ \ \exists x A \in \mathcal{F}.$$

Scope of the quantifier In $\forall x A,\ \exists x A$, A is in the scope of the quantifier $\forall,\ \exists$, respectively.

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

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

for #f = 1, #g = 2 , #R = 2. Some of the formulas of $\mathcal{L}$ are the following.

$$R(c, d), \quad \exists x R(x, f(c)), \quad \neg R(x, y), \quad (\exists x R(x, f(c)) \Rightarrow \neg R(x, y)),$$
$$(R(c, d) \cap \exists x R(x, f(c))), \quad \forall y R(y, g(c, g(x, f(c)))), \quad \forall y \neg \exists x R(x, y).$$

The formula $R(x, f(c))$ is in a scope of the quantifier $\exists x$ in $\exists x R(x, f(c))$. The formula $(\exists x R(x, f(c)) \Rightarrow \neg R(x, y))$ isn't in a scope of any quantifier. The formula $(\exists x R(x, f(c)) \Rightarrow \neg R(x, y))$ is in the scope of $\forall$ in $\forall z (\exists x R(x, f(c)) \Rightarrow \neg R(x, y))$.

Now we are ready to define formally a predicate language.

Predicate language Let $\mathcal{A}, T, \mathcal{F}$ be the alphabet, the set of terms and the set of formulas as defined above. A predicate language $\mathcal{L}$ is a triple

$$\mathcal{L} = (\mathcal{A}, T, \mathcal{F}).$$

As we have said before, the language $\mathcal{L}$ is determined by the choice of the symbols of its alphabet, namely of the choice of connectives, predicate, function, and constant symbols. If we want specifically mention this choice, we write

$$\mathcal{L} = \mathcal{L}_{CON}(\mathbf{P}, \mathbf{F}, \mathbf{C}) \ \text{ or } \ \mathcal{L} = \mathcal{L}(\mathbf{P}, \mathbf{F}, \mathbf{C}).$$

Given a predicate language $\mathcal{L} = (\mathcal{A}, T, \mathcal{F})$, we must distinguish between formulas like

$P(x, y), \quad \forall x P(x, y)$ and $\forall x \exists y P(x, y)$.

This is done by introducing the notion of free and bound variables, open and closed formulas (sentences).

Informally, in the formula

$$P(x, y)$$

both variables x and y are called *free* variables. They are not in the scope of any quantifier. The formula of that type (without quantifiers) is an open formula.

In the formula

$$\forall y P(x, y)$$

the variable x is free, the variable y is *bound* . The variable y is in the scope, is bounded by the quantifier $\exists$.

In the formula

$$\forall z P(x, y)$$

both x and y are free. In the formulas

$$\forall z P(z, y), \quad \forall x P(x, y)$$

only the variable y is free.

In the formula

$$\forall x(P(x) \Rightarrow \exists y Q(x, y))$$

there is no free variables, but in

$$(\forall x P(x) \Rightarrow \exists y Q(x, y))$$

the variable x (in $Q(x, y)$) is free.

Sometimes in order to distinguish more easily which variable is free and which is bound in the formula we might use the bold face type for the quantifier bound variables, i.e. to write the last formulas as

$$(\forall \mathbf{x} P(\mathbf{x}) \Rightarrow \exists \mathbf{y} Q(x, \mathbf{y})).$$

The formal definition of the set of free variables of a formula is the following.

Free variables The set $FV(A)$ of free variables of a formula A is defined by the induction of the degree of the formula as follows.

1. If A is an atomic formula, i.e. $A \in \mathcal{AF}$, then $FV(A)$ is just the set of variables appearing in the expression A;
2. for any unary propositional connective, i.e any $\bigtriangledown \in C_1$, $FV(\bigtriangledown A)$= $FV(A)$, i.e. the free variables of $\bigtriangledown A$ are the free variables of A;

3. for any binary propositional connective, i.e any $\circ \in C_2$,
   $FV(A \circ B) = FV(A) \cup FV(B)$,
   i.e. the free variables of $(A \circ B)$ are the free variables of A together with the free variables of B;
4. $FV(\forall x A) = FV(\exists x A) = FV(A) - \{x\}$,
   i.e. the free variables of $\forall x A$ and $\exists x A$ are the free variables of A, except for x.

Bound variables A variable is called bound if it is not free.

Sentence A formula with no free variables is called a sentence

Open formula A formula with no bound variables is called an open formula.

Example The formulas

$$\exists x Q(c, g(x,d)), \quad \neg \forall x (P(x) \Rightarrow \exists y (R(f(x), y) \cap \neg P(c)))$$

are sentences.

Example The formulas

$$Q(c, g(x,d)), \quad \neg (P(x) \Rightarrow (R(f(x), y) \cap \neg P(c)))$$

are open formulas.

Example The formulas

$$\exists x Q(c, g(x,y)), \quad \neg (P(x) \Rightarrow \exists y (R(f(x), y) \cap \neg P(c)))$$

are neither sentences nor open formulas. They contain some free and some bound variables; the variable y is free in the first formula, the variable x is free in the second.

It is common practice to use the notation

$$A(x_1, x_2, ..., x_n)$$

to indicate that $FV(A) \subseteq \{x_1, x_2, ..., x_n\}$ without implying that all of $x_1, x_2, ..., x_n$ are actually free in A. This is similar to the practise in algebra of writing $p(x_1, x_2, ..., x_n)$ for a polynomial p in the variables $x_1, x_2, ..., x_n$ without implying that all of them have nonzero coefficients.

Replacing x by t in A If $A(x)$ is a formula, and t is a term then

$$A(t/x)$$

or, more simply,

$$A(t)$$

denotes the result of replacing all occurrences of the free variable x by the term t throughout.

Notation When using the notation

$$A(t)$$

we always assume that none of the variables in t occur as bound variables in A.

The assumption that none of the variables in t occur as bound variables in A is essential, otherwise by substituting t on the place of x we would distort the meaning of $A(t)$.

Example Let $t = y$ and $A(x)$ is $\exists y(x \neq y)$, i.e. the variable y in t is bound in A. The substitution of t for x produces a formula $A(t)$ of the form $\exists y(y \neq y)$, which has a different meaning than $\exists y(x \neq y)$.

But if $t = z$, i.e. the variable z in t is not bound in A, then $A(t/x) = A(t)$ is $\exists y(z \neq y)$ and express the same meaning as $A(x)$.

Remark that if for example $t = f(z, x)$ we obtain $\exists y(f(z, x) \neq y)$ as a result of substitution of $t = f(z, x)$ for x in $\exists y(x \neq y)$.

This notation is convenient because we can agree to write as

$$A(t_1, t_2, ..., t_n) \quad \textit{or} \quad A(t_1/x_1, t_2/x_2, ..., t_n/x_n)$$

a result of substituting in A the terms $t_1, t_2, ..., t_n$ for all free occurrences (if any) of $x_1, x_2, ..., x_n$, respectively.

But when using this notation we always assume that none of the variables in $t_1, t_2, ..., t_n$ occur as bound variables in A.

The above assumption that none of the variables in $t_1, t_2, ..., t_n$ occur as bound variables in A is often expressed using the notion: $t_1, t_2, ..., t_n$ *are free for all theirs variables in* A which is defined formally as follows.

Term t is free for y in A

If $A \in \mathcal{F}$ and t is a term, then t is said to be free for y if no free occurrence of y lies within the scope of any quantifier bounding variables in t.

Example Let A , B be the formulas

$$\forall y P(f(x, y), y), \quad \forall y P(f(x, z), y),$$

respectively. The term $t = f(x, y)$ is free for x and is not free for y in A. The term $t = f(x, z)$ is free for x and z in B. The term $t = y$ is not free neither for x nor for z in A, B.

Example Let A be a formula

$$(\exists x Q(f(x), g(x,z)) \cap P(h(x,y), y)).$$

The term $t_1 = f(x)$ is not free for x in A; the term $t_2 = g(x,z)$ is free for z only, term $t_3 = h(x,y)$ is free for y only because x occurs as a bound variable in A; term t_4.

Notation If $A(x), A(x_1, x_2, ..., x_n) \in \mathcal{F}$ and $t, t_1, t_2, ..., t_n \in T$, then

$$A(t/x), \quad A(t_1/x_1, t_2/x_2, ..., t_n/x_n)$$

or, more simply just

$$A(t), \quad A(t_1, t_2, ..., t_n)$$

denotes the result of replacing all occurrences of the free variables x, $x_1, x_2, ..., x_n$, by the terms t, $t, t_1, t_2, ..., t_n$, respectively, assuming that $t, t_1, t_2, ..., t_n$ are free for all theirs variables in A.