Chapter 13: Predicate Languages

Predicate Languages are also 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.

$$0 < card(CON) < \aleph_0.$$

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

$$card(VAR) = \aleph_0.$$

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.

Observe that what really differ one predicate language from the other is the **choice of additional symbols** to the symbols described above.

These symbols predicate symbols, function symbols, and constant symbols.

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 **F** is **empty** we say that we deal with a language without functional symbols.

We denote functional symbols by $f, g, h, \ldots$, 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 **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 **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.

$$\mathbf{C} \subseteq \mathbf{F}.$$

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**.

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

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

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

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.

Terms The set T of all **terms** and

Formulas 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 1 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 2 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 For any predicate language $\mathcal{L}(\mathbf{P}, \mathbf{F}, \mathbf{C})$, the set T of its terms is always non-empty.

Example 3 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:

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

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

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.

Our language

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

is 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,

The language $\mathcal{L}$ 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

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

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

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

Another important notion of the Predicate language is the notion of **a scope** of the quantifier. It is defined as follows.

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.

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

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

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

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

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

The formula $(\exists x R(x, f(c)) \Rightarrow \neg R(x, x))$ **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 $\forall y (\exists x R(x, f(c)) \Rightarrow \neg R(x, y))$.

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.

Definition 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

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

Gentzen Style Proof System for Classical Predicate Logic - The System QRS

System QRS Definition

Let $\mathcal{F}$ denote a set of formulas of a Predicate (first Order) Logic Language

$$\mathcal{L}(\mathbf{P},\mathbf{F},\mathbf{C}) = \mathcal{L}_{\{\cap,\cup,\Rightarrow,\neg\}}(\mathbf{P},\mathbf{F},\mathbf{C})$$

for **P, F, C** countably infinite sets of predicate, functional, and constant symbols respectively.

The rules of inference of our system **QRS** will operate, as in the propositional case, on **finite sequences of formulas**, i.e. elements of $\mathcal{F}^*$, instead of just plain formulas $\mathcal{F}$, as in Hilbert style formalizations.

We will denote the sequences of formulas by Γ, Δ, Σ, with indices if necessary.

Intuitive semantics If Γ is a sequence

$$A_1, A_2, ..., A_n$$

then by δ_Γ we will understand the disjunction of all formulas of Γ.

As we know, the disjunction in classical logic is commutative, i.e., for any formulas A, B, C, $A \cup (B \cup C) \equiv (A \cup B) \cup C$, we will denote any of those formulas by

$$A \cup B \cup C = \delta_{\{A,B,C\}}.$$

Similarly, we will write

$$\delta_\Gamma = A_1 \cup A_2 \cup ... \cup A_n.$$

The sequence Γ is said to be **satisfiable** (**falsifiable**) if the formula $\delta_\Gamma = A_1 \cup A_2 \cup ... \cup A_n$ is satisfiable (falsifiable).

The sequence Γ is said to be **a tautology** if the formula $\delta_\Gamma = A_1 \cup A_2 \cup ... \cup A_n$ is a tautology.

The system QRS consists of **two axiom** schemas and **eleven rules** of inference.

The rules form two groups.

First group is similar to the propositional case and called

Each rule of this group introduces a new logical connective or its negation, so we will name them, as in the propositional case: $(\cup), (\neg\cup), (\cap), (\neg\cap), (\Rightarrow), (\neg \Rightarrow)$, and $(\neg\neg)$.

The second group deals with the quantifiers. It consists of four rules.

Two quantifiers rules introduce the universal and existential quantifiers, and are named $(\forall)$ and $(\exists)$, respectively.

The two other rules correspond to the De Morgan Laws and deal with the negation of the universal and existential quantifiers, and are named $(\neg\forall)$ and $(\neg\exists)$, respectively.

As the axioms we adopt any sequence which contains any formula and its negation, i.e any sequence of the form

$$\Gamma_1, A, \Gamma_2, \neg A, \Gamma_3$$

or of the form

$$\Gamma_1, \neg A, \Gamma_2, A, \Gamma_3,$$

for any formula $A \in \mathcal{F}$ and any sequences of formulas $\Gamma_1, \Gamma_2, \Gamma_3 \in \mathcal{F}^*$.

We will denote the axioms by

$$\mathcal{AX}^*.$$

QRS proof system is defined as

$$\mathbf{QRS} = (\mathcal{F}^*, \mathcal{AX}^*, \{(\cup), (\neg\cup), (\cap), (\neg\cap),$$

$$(\Rightarrow), (\neg \Rightarrow), (\neg\neg), (\neg\forall), (\neg\exists), (\forall), (\exists)\})$$

QRS system is called a **Gentzen- style formalization** of classical predicate calculus.

In order to define the rules of inference of **QRS** we need to introduce some definitions. They are straightforward modification of the corresponding definitions for the propositional logic.

We form, as in the propositional case, a special subset

$$\mathcal{LIT} \subseteq \mathcal{F}$$

of formulas, called a set of all **literals**, which is defined now as follows.

$$\mathcal{LIT} = \{A \in \mathcal{F} : A \in \mathcal{AF}\} \cup \{\neg A \in \mathcal{F} : A \in \mathcal{AF}\},$$

where $\mathcal{AF} \subseteq \mathcal{F}$ is the set of all **atomic (elementary) formulas** of the first order language, i.e.

$$\mathcal{AF} = \{P(t_1, ..., t_n) : P \in \mathbf{P}\ \}$$

$P \in \mathbf{P}$ is any n-argument predicate symbol, and $t_i \in T$ are terms.

The elements of the set

$$\{A \in \mathcal{F} : A \in \mathcal{AF}\}$$

are called **positive literals** and the elements of the set

$$\{\neg A \in \mathcal{F} : A \in \mathcal{AF}\}$$

are called **negative literals**.

I.e atomic (elementary) formulas are called positive literals and the negation of an atomic (elementary) formula is called a negative literal.

Indecomposable formulas Literals are also called the indecomposable formulas.

Now we form **finite sequences** out of formulas (and, as a special case, out of literals). We need to distinguish the sequences formed out of literals from the sequences formed out of other formulas, so we adopt exactly the same notation as in the propositional case.

We denote by $\Gamma^{'}$, $\Delta^{'}$, $\Sigma^{'}$ finite sequences (empty included) formed out of **literals** i.e. out of the elements of $\mathcal{LIT}$ i.e. we assume that

$$\Gamma^{'}, \Delta^{'}, \Sigma^{'} \in \mathcal{LIT}^*.$$

We denote by Γ, Δ, Σ the elements of $\mathcal{F}^*$ i.e the finite sequences (empty included) formed out of elements of $\mathcal{F}$.

We define the inference rules of **QRS** as follows.

Group 1: Propositional Inference rules

Disjunction rules

$$(\cup)\ \frac{\Gamma', A, B, \Delta}{\Gamma', (A \cup B), \Delta}, \qquad (\neg\cup)\ \frac{\Gamma', \neg A, \Delta \ \ : \ \ \Gamma', \neg B, \Delta}{\Gamma', \neg(A \cup B), \Delta}$$

Conjunction rules

$$(\cap)\ \frac{\Gamma', A, \Delta \ ; \ \Gamma', B, \Delta}{\Gamma', (A \cap B), \Delta}, \qquad (\neg\cap)\ \frac{\Gamma', \neg A, \neg B, \Delta}{\Gamma', \neg(A \cap B), \Delta}$$

Implication rules

$$(\Rightarrow)\ \frac{\Gamma', \neg A, B, \Delta}{\Gamma', (A \Rightarrow B), \Delta}, \qquad (\neg \Rightarrow)\ \frac{\Gamma', A, \Delta \ \ : \ \ \Gamma', \neg B, \Delta}{\Gamma', \neg(A \Rightarrow B), \Delta}$$

Negation rule

$$(\neg\neg)\ \frac{\Gamma', A, \Delta}{\Gamma', \neg\neg A, \Delta}$$

where $\Gamma' \in \mathcal{F}^*, \Delta \in \mathcal{F}'^*, A, B \in \mathcal{F}$.

Group 2: Quantifiers Rules

($\exists$)

$$\frac{\Gamma', A(t), \Delta, \exists x A(x)}{\Gamma', \exists x A(x), \Delta}$$

where t is an arbitrary term.

($\forall$)

$$\frac{\Gamma', A(y), \Delta}{\Gamma', \forall x A(x), \Delta}$$

where y is a free individual variable which does not appear in any formula in the conclusion, i.e. in the sequence $\Gamma', \forall x A(x), \Delta$.

The variable y in ($\forall$) is called the **eigenvariable**.

The condition: *where* y *is a free individual variable which does not appear in any formula in the conclusion* is called the **eigenvariable condition**.

All occurrences of y in $A(y)$ of the rule $(\forall)$ are fully indicated.

($\neg\forall$)

$$\frac{\Gamma', \exists x \neg A(x), \Delta}{\Gamma', \neg\forall x A(x), \Delta}$$

($\neg\exists$)

$$\frac{\Gamma', \forall x \neg A(x), \Delta}{\Gamma', \neg\exists x A(x), \Delta}$$

$\Gamma' \in \mathcal{LIT}^*, \Delta \in \mathcal{F}^*, A, B \in \mathcal{F}$.

Note that $A(t), A(y)$ denotes a formula obtained from $A(x)$ by writing t, y, respectively, in place of all occurrences of x in A.

We define the notion of a *formal proof* in **QRS** as in any proof system, i.e., by a formal proof of a sequence Γ in the proof system **QRS** we understand any sequence

$$\Gamma_1; \Gamma_2; ....\Gamma_n$$

of sequences of formulas (elements of $\mathcal{F}^*$, such that $\Gamma_1 \in \mathcal{AX}^*$, $\Gamma_n = \Gamma$, and for all i $(1 < i \leq n)$ $\Gamma_i \in \mathcal{AX}^*$, or Γ_i is a conclusion of one of the inference rules of **QRS** with all its premisses placed in the sequence $\Gamma_1; \Gamma_2; ....\Gamma_{i-1}$.

As the proof system under consideration is fixed, we will write, as usual,

$$\vdash \Gamma$$

to denote that Γ has a formal proof in **QRS**.

Given a formula $A \in \mathcal{F}$, we define its decomposition tree $\mathcal{T}_A$ in a similar way as in the propositional case.

The decomposition tree of the de Morgan Law $(\neg\forall x A(x) \Rightarrow \exists x \neg A(x))$ is the following.

$$(\neg\forall x A(x) \Rightarrow \exists x \neg A(x))$$

$$\mid (\Rightarrow)$$

$$\neg\neg\forall x A(x), \exists x \neg A(x)$$

$$\mid (\neg\neg)$$

$$\forall x A(x), \exists x \neg A(x)$$

$$\mid (\forall)$$

$$A(x_1), \exists x \neg A(x)$$

where x_1 is a first free variable in the sequence such that x_1 does not appear in $\forall x A(x), \exists x \neg A(x)$

$$\mid (\exists)$$

$$A(x_1), \neg A(x_1), \exists x \neg A(x)$$

where x_1 is the first term (variables are terms) in the sequence such that $\neg A(x_1)$ does not appear on a tree above $A(x_1), \neg A(x_1), \exists x \neg A(x)$

Axiom