LOGICS FOR COMPUTER SCIENCE: Classical and Non-Classical

Anita Wasilewska

Chapter 8
Classical Predicate Semantics and Proof Systems

CHAPTER 8 SLIDES

Slides Set 1

Chapter 8
Classical Predicate Semantics and Proof Systems

Slides Set 1

PART 1: Formal Predicate Languages

Slides Set 2

PART 2: Classical Semantics

Chapter 8
Classical Predicate Semantics and Proof Systems

Slides Set 3

PART 3: Predicate Tautologies, Equational Laws of Quantifiers

PART 4: Proof Systems: Soundness and Completeness

Chapter 8
Classical Predicate Semantics and Proof Systems

Slides Set 1

PART 1: Formal Predicate Languages

Formal Predicate Languages

We define a **predicate** language $\mathcal{L}$ following the pattern established by the propositional languages

The **predicate** language $\mathcal{L}$ is more complicated in its structure and hence its **alphabet** $\mathcal{A}$ is much richer

The definition of its set $\mathcal{F}$ of **formulas** is more complicated

In order to define the set $\mathcal{F}$ of formulas we introduce an additional set **T**, called a set of **terms**

The **terms** play important role in the development of other notions of **predicate** logic

Predicate Languages

Predicate languages are also called first order languages

The same applies to the use of terms for propositional and predicate logics

Propositional and **predicate** logics are called zero order and first order logics, respectively

We will use both terms **equally**

We work with many different **predicate** languages, depending on what applications we have in mind

All of these **languages** have some common features, and we begin with a following general definition

Predicate Language

Definition

By a **predicate language** $\mathcal{L}$ we understand a triple

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

where

$\mathcal{A}$ is a predicate **alphabet**

$\mathbf{T}$ is the set of **terms**

$\mathcal{F}$ is a set of **formulas**

Predicate Languages Components

The first **component** of $\mathcal{L}$ is defined as follows

1. Alphabet $\mathcal{A}$ is the set

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

where

VAR is set of **predicate variables**

CON is a set of **propositional connectives**

PAR is a set of **parenthesis**

Q is a set of **quantifiers**

P is a set of **predicate** symbols

F i a set of **functions** symbols, and

C is a set of **constant** symbols

We assume that all of the sets defining the alphabet are **disjoint**

Alphabet Components

The **component** of the **alphabet** $\mathcal{A}$ are defined as follows

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. we often express it by writing

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

Alphabet Components

Propositional Connectives

We define the set of propositional connectives CON in the same way as in the propositional case

The set CON 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

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

Alphabet Components

The set of propositional connectives CON defines a propositional part of the **predicate** language

What really **differs** one predicate language from the other is the choice of the following additional symbols

These are quantifiers symbols, predicate symbols, function symbols, and constant symbols

A particular **predicate** language is determined by **specifying** the following **sets** of symbols of the alphabet

Alphabet Components

Quantifiers

We adopt two quantifiers;

universal quantifier denoted by $\forall$ and

existential quantifier denoted by $\exists$

We have the following set of quantifiers

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

Alphabet Components

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** of quantifiers is impossible in a case of some non-classical logics, for example for the intuitionistic logic

But even in the case of **classical** logic we often adopt the two quantifiers as they express better the intuitive understanding of formulas

Alphabet Components

Predicate symbols

Predicate symbols **represent** relations

Any **predicate** language contains a non empty, finite or countably infinite set

$$\mathbf{P}$$

of **predicate** symbols. We **denote** predicate symbols by

$$P, Q, R, \ \ldots$$

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** symbol

Alphabet Components

Function symbols

Function symbols **represent** functions

Any **predicate** language contains a finite (may be empty) or countably infinite set

$$\mathbf{F}$$

of **function** symbols. We **denote** functional symbols by

$$f, g, h, \ \ldots$$

with indices, if necessary

When $\mathbf{F} = \emptyset$ we say that we deal with a language **without** functional symbols

Each **function** symbol $f \in \mathbf{F}$ has a positive integer $\#f$ assigned to it

if $\#f = n$ then f is called an n-ary (n - place) **function symbol**

Alphabet Components

Constant symbols

Any **predicate** language contains a finite (may be empty) or countably infinite set

$$\mathbf{C}$$

of **constant** symbols

The elements of $\mathbf{C}$ are **denoted** by

$$c, d, e, \ \ldots$$

with indices, if necessary

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

We single them out as a separate set for our convenience

Predicate Language

Given an **alphabet**

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

What distinguishes one **predicate** language

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

from the other is the **choice** of the components *CON* and the sets **P, F, C** of its alphabet $\mathcal{A}$

We hence will write

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

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

Predicate Language Notation

Once the set *CON* of propositional connectives is **fixed**, the predicate language

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

is determined by the sets **P, F** and **C**

We write

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

for the predicate language $\mathcal{L}$ determined by **P, F,C** (with a fixed set of propositional connectives)

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

Predicate Languages Notation

We sometimes allow the same symbol to be used as an n-place predicate 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

Predicate Language

Having defined the **basic** element of syntax, the **alphabet** $\mathcal{A}$, we can now complete the formal definition of the predicate language

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

by defining next **two** more complex components:

the set $\mathbf{T}$ of all **terms** and

the set $\mathcal{F}$ of all well formed **formulas** of the language $\mathcal{L} = \mathcal{L}_{CON}(\mathbf{P}, \mathbf{F}, \mathbf{C})$

Set of Terms

Terms

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

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

meeting the conditions:

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

Terms Examples

Example 1

Let $f \in \mathbf{F}$, $\#f = 1$, i.e. f is a 1-place **function symbol**

Let x, y be **variables**, c, d be **constants**, i.e.

$$x, y \in VAR \quad \text{and} \quad c, d \in \mathbf{C}$$

Then the following expressions are **terms**:

$$x, \quad y, \quad f(x), \quad f(y), \quad f(c), \quad f(d), \ \ldots$$

$$f(f(x)), \quad f(f(y)), \quad f(f(c)), \quad f(f(d)), \ \ldots$$

$$f(f(f(x))), \quad f(f(f(y))), \quad f(f(f(c))), \quad f(f(f(d))), \ \ldots$$

Terms Examples

Example 2

Let $\mathbf{F} = \emptyset, \mathbf{C} = \emptyset$

In this case terms consists of **variables only**, i.e.

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

Directly from the **Example 2** we get the following

Remark

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

Terms Examples

Example 3

Consider a case of $\mathcal{L}(\mathbf{P}, \mathbf{F}, \mathbf{C})$ where

$$\mathbf{F} = \{\ f,\ g\ \} \quad \text{for} \quad \#f = 1 \ \text{ and } \ \#g = 2$$

Let $x, y \in VAR$ and $c,\ d \in \mathbf{C}$

Some of the **terms** are the following:

$$f(g(x, y)), \quad f(g(c, x)), \quad g(f(f(c)), g(x, y)),$$
$$g(c, g(x, f(c))), \quad g(f(g(x, y)), g(x, f(c))), \ \ldots$$

Terms Notation

From time to time, the logicians are and so we may be also informal about the way we write terms

Example

If we **denote** a 2- place function symbol g by $+$, we may **write**

$$x + y \quad \text{instead of writing} \quad +(x, y)$$

Because in this case we can **think** of $x + y$ as an unofficial way of designating the ”real” **term** $g(x, y)$

Atomic Formulas

Atomic Formulas

Before we define formally the set $\mathcal{F}$ of **formulas**, we need to define one more set, namely the set of **atomic**, or **elementary** formulas

Atomic formulas are the simplest formulas

They building blocks for other formulas the way the propositional variables were in the case of **propositional** languages

Atomic Formulas

Definition

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 $R \in \mathbf{P}$, $\#R = n$ and $t_1, t_2, ..., t_n \in \mathbf{T}$

I.e. R is n-ary predicate (relational) symbol and $t_1, t_2, ..., t_n$ are any terms

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

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

Atomic Formulas Examples

Example

Consider a language

$$\mathcal{L} = \mathcal{L}(\{P\}, \emptyset, \emptyset) \quad \text{for} \quad \#P = 1$$

$\mathcal{L}$ is a predicate language **without** neither functional, nor constant symbols, and with only **one**, 1-place predicate symbol P

The set $\mathcal{AF}$ of **atomic** formulas contains all formulas of the form $P(x)$, for x any variable, i.e.

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

Atomic Formulas Examples

Example

Let now consider a **predicate language**

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

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 2-place **predicate symbol** R, and two **constants:** c,d

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

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

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

Set of Formulas Definition

Set $\mathcal{F}$ of Formulas

Given a predicate language

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

where *CON* is *non-empty*, *finite set* of propositional connectives such that $CON = C_1 \cup C_2$ for C_1 a finite set (possibly empty) of unary connectives, C_2 a finite set (possibly empty) of binary connectives of the language $\mathcal{L}$

We define the set $\mathcal{F}$ of all well formed formulas of the predicate language $\mathcal{L} = \mathcal{L}_{CON}(\mathbf{P}, \mathbf{F}, \mathbf{C})$ as follows

Set of Formulas Definition

Definition

The set $\mathcal{F}$ of all well formed **formulas**, of the language $\mathcal{L} = \mathcal{L}_{CON}(\mathbf{P}, \mathbf{F}, \mathbf{C})$ is the **smallest** set meeting the following conditions

1. Any **atomic formula** of $\mathcal{L}$ is a **formula** , i.e.

$$A\mathcal{F} \subseteq \mathcal{F}$$

2. If A is a formula of $\mathcal{L}$, $\triangledown$ is an one argument **propositional connective**, then $\triangledown A$ is a formula of $\mathcal{L}$, 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 $\mathcal{L}$ and $\circ$ is a two argument **propositional connective**, then $(A \circ B)$ is a **formula** of $\mathcal{L}$, i.e. the following **recursive condition** holds

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

4. If A is a **formula** of $\mathcal{L}$ and x is a **variable**, $\forall, \exists \in \mathbf{Q}$, then $\forall xA, \ \exists xA$ are **formulas** of $\mathcal{L}$, i.e. the following recursive condition holds

$$\text{If} \quad A \in \mathcal{F}, \ x \in VAR, \quad \forall, \exists \in \mathbf{Q}, \quad \text{then} \quad \forall xA, \ \exists xA \in \mathcal{F}$$

Scope of Quantifiers

Another important notion of the predicate language is the notion of scope of a quantifier

Definition

Given formulas

$$\forall xA, \quad \exists xA$$

The formula A is said to be in the **scope** of a 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.

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

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

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

Scope of Quantifiers

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

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

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

Scope of Quantifiers

Example

Let $\mathcal{L}$ be a first order language of some **modal** logic defined as follow

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

where

$$\#f = 1, \quad \#g = 2, \quad \#R = 2$$

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

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

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

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

$$\Box\forall y\neg\Diamond\exists x R(x, y)$$

Scope of Quantifiers

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

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 the quantifier $\forall$ in $\forall z(\exists x R(x, f(c)) \Rightarrow \neg R(x, y))$

Formula $\neg\Diamond\exists x R(x, y)$ is in the **scope** of the quantifier $\forall$ in $\Box\forall y\neg\Diamond\exists x R(x, y)$

Free and Bound Variables

Given a predicate language $\mathcal{L} = (\mathcal{A}, T, \mathcal{F})$

We want to distinguish between formulas like

$$P(x, y), \quad \forall x P(x, y) \quad \text{and} \quad \forall x \exists y P(x, y)$$

This is done by introducing the notion of free and bound **variables** as well as the notion of open and closed **formulas** (sentences)

Before we formulate proper definitions, here are some simple observations

Free and Bound Variables

1. Some formulas are **without** quantifiers

For example formulas

$$R(c_1, c_2), \quad R(x, y), \quad (R(y, d) \Rightarrow R(a, z))$$

Variables x, y in $R(x, y)$ are called **free** variables

The variables y in R(y, d), and z in R(a,z) are also **free**

A formula **without** quantifiers is called an **open** formula

Free and Bound Variables

2. Quantifiers **bind** variables within formulas

In the formula

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

the variable x is **free**, the variable y is **bounded** by the the quantifier $\forall$

In the formula

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

both x and y are **free**

In both formulas

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

only the variable y is **free**

Free and Bound Variables

3. The formula $\exists x \forall y R(x, y)$ **does not** contain any free variables, neither does the formula $R(c_1, c_2)$

A formula without any free variables is called called a **closed** formula or a **sentence**

The formula

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

is a **closed** formula (**sentence**), the formula

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

is not a sentence

Free and Bound Variables

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 and write the formulas as follows

$$(\forall \mathbf{x} Q(\mathbf{x}, y), \quad \exists \mathbf{y} P(\mathbf{y}), \quad \forall \mathbf{y} R(\mathbf{y}, g(c, g(x, f(c)))),$$

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

Observe that the formulas

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

are **sentences**

Free and Bound Variables Formal Definition

Definition

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. $A \in A\mathcal{F}$, then $FV(A)$ is just the set of variables appearing in A;
2. for any **unary** propositional connective, i.e. for any $\triangledown \in C_1$

$$FV(\triangledown A) = FV(A)$$

   i.e. the **free** variables of $\triangledown A$ are the **free** variables of A;
3. for any **binary** propositional connective, i.e, for 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. $FV(\forall xA) = FV(\exists xA) = FV(A) - \{x\}$
   i.e. the **free** variables of $\forall xA$ and $\exists xA$ are the **free** variables of A, **except** for x

Important Notation

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 practice in **algebra** of writing $w(a_0, a_1, ..., a_n) = a_0 + a_1 x + ... + a_n x^n$ for a polynomial w without implying that **all** of the coefficients $a_0, a_1, ..., a_n$ are nonzero

Replacements

Replacing x **by** t **in** Ax

Given a formula $A(x)$ and a term t. We denote by

$$A(x/t) \text{ or simply by } A(t)$$

the result of **replacing** all occurrences of the free variable x in A by the **term** t

When performing the **replacement** we always assume that **none** of the variables in t occur as bound variables in A

Replacement

Reminder

When **replacing** a variable x by a term $t \in \mathbf{T}$ in a formula $A(x)$, we denote the result as

$$A(t)$$

We do it under the assumption 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 $A(t)$ is essential because **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 $t = y$ for the variable x produces a formula $A(t)$ of the form

$$\exists y(y \neq y)$$

which has a **different meaning** than

$$\exists y(x \neq y)$$

Example

Let now $t = z$ and the formula $A(x)$ is

$$\exists y(x \neq y)$$

i.e. the variable z in t **is not bound** in A

The substitution of $t = z$ for the variable x produces a formula $A(t)$ of the form

$$\exists y(z \neq y)$$

which express the **same meaning** as $A(x)$

Special Terms

Here an important notion we will depend on

Definition

Given $A \in \mathcal{F}$ and $t \in \mathbf{T}$

The **term** t is said to be **free for** a variable x in a formula A

if and only if

no free occurrence of x **lies** within the **scope** of any quantifier bounding variables in t

Special Terms

Example

Given formulas

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

The term $t = f(x,y)$ is **free** for x in $\forall y P(f(x,y),y)$ and $t = f(x,y)$ is **not free** for y in $\forall y P(f(x,y),y)$

The term

$$t = f(x,z)$$

is **free** for x and z in

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

Special Terms

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

Replacement Definition

Given

$$A(x),\quad A(x_1, x_2, ..., x_n) \in \mathcal{F} \quad \text{and} \quad t, t_1, t_2, ..., t_n \in \mathbf{T}$$

Then

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

or, more simply just

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

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

Classical Restricted Domain Quantifiers

Restricted Domain Quantifiers

We often use logic **symbols**, while writing mathematical statements

For example, mathematicians in order to say

”all natural numbers are greater then zero and some integers are equal 1”

often write it as

$$x \geq 0, \forall_{x \in N} \text{ and } \exists_{y \in Z},\ y = 1$$

Some of them, who are more ”logic oriented”, would also write it as

$$\forall_{x \in N}\, x \geq 0 \ \cap \ \exists_{y \in Z}\, y = 1$$

or even as

$$(\forall_{x \in N}\, x \geq 0 \ \cap \ \exists_{y \in Z}\, y = 1)$$

Restricted Domain Quantifiers

None of the above symbolic statements are **formulas** of the predicate language $\mathcal{L}$

These are **mathematical** statement written with mathematical and logic **symbols**

They are written with different **degree** of ”logical precision” , the last being, from a logician point of view the most **precise**

Restricted Domain Quantifiers

Observe that the quantifiers symbols

$$\forall_{x \in N} \quad \text{and} \quad \exists_{y \in Z}$$

used in all of the symbolic mathematical statements **are not** the one used in the **predicate** language $\mathcal{L}$

The quantifiers of this type are called quantifiers with **restricted domain**

Our **goal** now is to correctly "translate " mathematical and natural language statement into well formed **formulas** of the predicate language

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

of the classical predicate logic

Restricted Domain Quantifiers

We say

" **formulas** of the predicate language $\mathcal{L}$ of the classical predicate logic"

to express the **fact** that we define all notions for the classical semantics

One can extend these definitions to some non-classical logics, but we describe and will investigate only the classical case

Restricted Domain Quantifiers

We introduce the quantifiers with **restricted domain** by expressing them **within** the predicate language $\mathcal{L}_{\{\neg.\cap,\cup,\Rightarrow\}}(\mathbf{P}, \mathbf{F}, \mathbf{C})$ as follows

Given a classical predicate logic language

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

The quantifiers

$$\forall_{A(x)} \quad \text{and} \quad \exists_{A(x)}$$

are called quantifiers with **restricted domain**, or **restricted quantifiers**, where $A(x) \in \mathcal{F}$ is any formula with any free variable $x \in VAR$

Restricted Domain Quantifiers

Definition

A formula $\forall_{A(x)} B(x)$ is an **abbreviation** of a formula $\forall x(A(x) \Rightarrow B(x)) \in \mathcal{F}$

We write it symbolically as

$$(*) \quad \forall_{A(x)}\, B(x) = \forall x(A(x) \Rightarrow B(x))$$

A formula $\exists_{A(x)} B(x)$ is an **abbreviation** of a formula $\exists x(A(x) \cap B(x)) \in \mathcal{F}$

We write it symbolically as

$$(**) \quad \exists_{A(x)}\, B(x) = \exists x(A(x) \cap B(x))$$

We call $(*)$ and $(**)$ the **transformations rules** for restricted quantifiers

Exercise

Exercise

Given the following mathematical statement **S** written with logical symbols

$$(\forall_{x \in N}\, x \geq 0 \;\cap\; \exists_{y \in Z}\, y = 1)$$

1. Translate the statement **S** into a proper logical **formula** A that uses **restricted** quantifiers

2. Translate the obtained **restricted quantifiers** formula A into a correct logical formula **without** restricted domain quantifiers, i.e. into a well formed formula of $\mathcal{L}$

Translation Steps

Given a mathematical statement **S**

We proceed to **write** this and other similar problems **translation** in a sequence of the following steps

Step 1

We identify **basic** statements in **S** i.e. mathematical statements that involve only **relations**

They are to be translated into **atomic formulas**

We identify the **relations** in the basic statements and choose **predicate** symbols as their names

We identify all **functions** and **constants** (if any) in the basic statements and choose **function** symbols and **constant** symbols as their names

Translation Steps

Step 2

We write the basic statements as **atomic** formulas of $\mathcal{L}$

Step 3

We re-write the statement **S** as a logical **formula** with restricted quantifiers

Step 4

We apply the transformations rules $(*)$ and $(**)$ for restricted quantifiers to the formula from **Step 3**

Such obtained **formula** A of $\mathcal{L}$ is a representation, which we call a **translation**, of the given mathematical statement **S**

Exercise Solution

Solution

The mathematical statement **S** is

$$(\forall_{x \in N}\, x \geq 0 \ \cap\ \exists_{y \in Z}\, y = 1)$$

Step 1 in this particular case is as follows

The basic statements in **S** are

$$x \in N, \quad x \geq 0, \quad y \in Z, \quad y = 1$$

The relations are $\in N, \quad \in Z, \quad \geq, \quad =$

We use one argument **predicate** symbols N, Z for relations $\in N$, $\in Z$, respectively

We use two argument **predicate** symbol G for $\geq$

We use predicate symbol E for $=$

There are **no functions**

We have two **constant** symbols c_1, c_2 for numbers 0 and 1, respectively

Exercise Solution

Step 2

We write $N(x), Z(x)$ for $x \in N$, $x \in Z$, respectively

We write $G(x, c_1)$ for $x \geq 0$ and $E(y, c_2)$ for $y = 1$

Atomic formulas are

$$N(x), \quad Z(x), \quad G(x, c_1), \quad E(y, c_2)$$

Step 3

The statement **S** becomes a **restricted quantifiers** formula

$$(\forall_{N(x)} G(x, c_1) \cap \exists_{Z(y)} E(y, c_2))$$

Step 4

A formula $A \in \mathcal{F}$ that is a a **translation** of **S** is

$$(\forall x\, (N(x) \Rightarrow G(x, c_1)) \cap \exists y\, (Z(y) \cap E(y, c_2)))$$

Exercise Short Solution

Here is a perfectly acceptable short solution

We presented first the long solution in order to **explain** in detail how one approaches the " translations " problems

This is why we identified the **Steps 1 - 4** needed to be performed when one does the **translation**

We use the word **translation** a short cut for saying

" The **formula** A is a formal predicate language $\mathcal{L}$ **representation** of the given mathematical statement **S**"

Short Solution

The basic statements in **S** are

$$x \in N, \quad x \geq 0, \quad y \in Z, \quad y = 1$$

The corresponding **atomic** formulas of $\mathcal{L}$ are

$$N(x), \quad Z(x), \quad G(x, c_1), \quad E(y, c_2)$$

The statement **S** becomes a **restricted quantifiers** formula

$$(\forall_{N(x)} G(x, c_1) \cap \exists_{Z(y)} E(y, c_2))$$

A formula $A \in \mathcal{F}$ that is a a **translation** of **S** is

$$(\forall x \, (N(x) \Rightarrow G(x, c_1)) \cap \exists y \, (Z(y) \cap E(y, c_2)))$$