LOGICS FOR COMPUTER SCIENCE: Classical and Non-Classical Springer 2019

Anita Wasilewska

Chapter 8
Classical Predicate Semantics and Proof Systems

CHAPTER 8 SLIDES

Chapter 8
Classical Predicate Semantics and Proof Systems

Slides Set 1

PART 1: Formal Predicate Languages

Slides Set 2

PART 2: Classical Semantics

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

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 **C** are **denoted** by

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

with indices, if necessary

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

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

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

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

Scope of Quantifiers

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

Definition

Given formulas

$$\forall x A, \quad \exists x A$$

The formula A is said to be in the **scope** of a quantifier $\forall, \exists$, respectively.

Scope of Quantifiers

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 the language $\mathcal{L}$ 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

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

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**"

Exercise Short Solution

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

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

Chapter 8
Classical Predicate Semantics and Proof Systems

Slides Set 2

PART 2: Classical Semantics

Classical Semantics

The notion of **predicate tautology** is much more **complicated** then that of the propositional

Predicate tautologies are also called **valid** formulas, or **laws of quantifiers** to distinguish them from the propositional case

The formulas of a predicate language $\mathcal{L}$ have meaning only when an **interpretation** is given for all its symbols

Classical Semantics

We define an **interpretation** I by interpreting predicate and functional symbols as a concrete **relation** and **function** defined in a certain set $U \neq \emptyset$

Constants symbols are interpreted as **elements** of the set U

The set U is called the **universe** of the interpretation I

These two items specify a **structure**

$$\mathbf{M} = (U, I) \quad \text{for the language} \quad \mathcal{L}_{CON}(\mathbf{P}, \mathbf{F}, \mathbf{C})$$

Classical Semantics

The **semantics** for a first order (predicate) language $\mathcal{L}$ in general, and for the first order classical logic in particular, is **defined**, after Tarski (1936), in terms of

the **structure** $\mathbf{M} = [U, I]$

an **assignment** s of $\mathcal{L}$

a **satisfaction relation** $(\mathbf{M}, s) \models A$ between structures, assignments and formulas of $\mathcal{L}$

The definition of the structure $\mathbf{M} = [U, I]$ and the assignment s of $\mathcal{L}$ is **common** for different predicate languages and for different semantics and we define them as follows.

Structure Definition

Definition

Given a predicate language

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

A **structure** for $\mathcal{L}$ is a pair

$$\mathbf{M} = [U, I]$$

where U is a non empty set called a **universe**

I is an assignment called an **interpretation** of the language $\mathcal{L}(\mathbf{P}, \mathbf{F}, \mathbf{C})$ in the universe U

The structure $\mathbf{M} = [U, I]$ components are defined as follows

Structure Definition

Structure $\mathbf{M} = [U, I]$ **Components**

1. I assigns to any predicate symbol $P \in \mathbf{P}$ a **relation** P_I defined in the universe U, i.e. for any $P \in \mathbf{P}$, if $\#P = n$, then

$$P_I \subseteq U^n$$

2. I assigns to any functional symbol $f \in \mathbf{F}$ a **function** f_I defined in the universe U, i.e. for any $f \in \mathbf{F}$, if $\#f = n$, then

$$f_I : \quad U^n \longrightarrow U$$

3. I assigns to any constant symbol $c \in \mathbf{C}$ an **element** c_I of the universe, i.e for any $c \in \mathbf{C}$,

$$c_I \in U$$

Structure Example

Example

Let $\mathcal{L}$ be a language with one two-place predicate symbol, two functional symbols: one -place and one two-place, and two constants, i.e.

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

where #R = 2, #f = 1, #g = 2, and $c, d \in$ **C**

We define a **structure** **M** = [U, I] as follows

We take as the universe the set $U = \{1, 3, 5, 6\}$

The **predicate** R is interpreted as $\leq$ what we write as

$$R_I : \leq$$

Structure Example

We interpret f as a **function** $f_I : \{1,3,5,6\} \longrightarrow \{1,3,5,6\}$ such that

$$f_I(x) = 5 \quad \text{for all} \quad x \in \{1,3,5,6\}$$

We put $g_I : \{1,3,5,6\} \times \{1,3,5,6\} \longrightarrow \{1,3,5,6\}$ such that

$$g_I(x,y) = 1 \quad \text{for all} \quad x \in \{1,3,5,6\}$$

The constant c becomes $c_I = 3$, and $d_I = 6$

We write the structure **M** as

$$\mathbf{M} = [\{1,3,5,6\} \quad \leq, \quad f_I, \quad g_I, \quad c_I = 3, \quad d_I = 6]$$

Assignment - Interpretation of Variables

Definition

Given a first order language

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

with the set VAR of variables

Let $\mathbf{M} = [U, I]$ be a structure for $\mathcal{L}$ with the universe $U \neq \emptyset$

An **assignment of** $\mathcal{L}$ **in** $\mathbf{M} = [U, I]$ is any function

$$s : \quad VAR \longrightarrow U$$

The **assignment** s is also called an **interpretation of variables** VAR of $\mathcal{L}$ in the structure $\mathbf{M} = [U, I]$

Assignment - Interpretation

Let $\mathbf{M} = [U, I]$ be a structure for $\mathcal{L}$ and

$$s : VAR \longrightarrow U$$

be an **assignment** of variables VAR of $\mathcal{L}$ in the structure $\mathbf{M}$

Let $\mathbf{T}$ be the set of all **terms** of $\mathcal{L}$

By definition of terns

$$VAR \subseteq \mathbf{T}$$

We use the interpretation I of the structure $\mathbf{M} = [U, I]$ to **extend** the **assignment** s to the set the set $\mathbf{T}$ of all **terms** of the language $\mathcal{L}$

Interpretation of Terms

Notation

We denote the **extension** of the assignment s to the set $\mathbf{T}$ by s_I rather then by s^* as we did before

s_I associates with each term $t \in \mathbf{T}$ an element $s_I(t) \in U$ of the universe of the structure $\mathbf{M} = [U, I]$

We **define** the extension s_I of s by the induction of the length of the term $t \in \mathbf{T}$ and call it an **interpretation of terms** of $\mathcal{L}$ in a structure $\mathbf{M} = [U, I]$

Interpretation of Terms

Definition

Given a language $\mathcal{L} = \mathcal{L}(\mathbf{P}, \mathbf{F}, \mathbf{C})$ and a structure $\mathbf{M} = [U, I]$

Let a function

$$s : VAR \longrightarrow U$$

be any assignment of variables VAR of $\mathcal{L}$ in $\mathbf{M}$

We **extend** s to a function

$$s_I : \mathbf{T} \longrightarrow U$$

called an **interpretation of terms** of $\mathcal{L}$ in $\mathbf{M}$

Interpretation of Terms

We define the function s_I by induction on the complexity of terms as follows

1. For any v $x \in VAR$,

$$s_I(x) = s(x)$$

2. for any $c \in \mathbf{C}$,

$$s_I(c) = c_I;$$

3. for any $t_1, \ t_2, \ \ldots, t_n \in \mathbf{T}, \ \ n \geq 1, \ \ f \in \mathbf{F}$, such that $\#f = n$

$$s_I(f(t_1, \ t_2, \ \ldots, t_n)) = f_I(s_I(t_1), \ s_I(t_2), \ \ldots, \ s_I(t_n))$$

Interpretation of Terms Example

Example

Consider a language

$$\mathcal{L} = \mathcal{L}(\{P,\ R\},\ \{f, h\},\ \emptyset$$

for # P = # R = 2, #f = 1, # h= 2

Let $\mathbf{M} = [Z, I\,]$, where Z is the set on integers and the **interpretation** I for elements of **F** and **C** is as follows

$f_I : Z \longrightarrow Z$ is given by formula $f(m) = m + 1$ for all $m \in Z$

$h_I : Z \times Z \longrightarrow Z$ is given by formula $f(m, n) = m + n$ for all $m, n \in Z$

Interpretation of Terms Example

Let s be any assignment $s : VAR \longrightarrow Z$ such that

$s(x) = -5, \;\; s(y) = 2$ and $t_1, t_2 \in \mathbf{T}$

Let $t_1 = h(y, f(f(x)))$ and $t_2 = h(f(x), h(x, f(y))$

We **evaluate**

$$s_I(t_1) = s_I(h(y, f(x)) \; = \; h_I(s_I(y), f_I(s_I(x))) \; =$$

$$+(2, f_I(-5)) = 2 - 4 = -2$$

and

$$s_I(t_2) = s_I(h(f(x), h(x, f(y))) =$$

$$+(f_I(-5), +(-5, 3)) = -4 + (-5 + 3) = -6$$

Observation

Given $t \in \mathbf{T}$

Let $x_1, x_2, \ldots, x_n \in VAR$ be all variables appearing in t

We write it as

$$t(x_1, x_2, \ldots, x_n)$$

Observation

For any term $t(x_1, x_2, \ldots, x_n) \in \mathbf{T}$, any structure $\mathbf{M} = [U, I]$ and any assignments s, s' of $\mathcal{L}$ in $\mathbf{M}$, the following holds

If $s(x) = s'(x)$ for all $x \in \{x_1, x_2, \ldots, x_n\}$, i.e

if the assignments s, s' **agree** on all variables appearing in t,

then

$$s_I(t) = s'_I(t)$$

Notation

Thus for any $t \in \mathbf{T}$, the function $s_I : \mathbf{T} \longrightarrow U$ **depends** on only a **finite** number of values of $s(x)$ for $x \in VAR$

Notation

Given a structure $\mathbf{M} = [U, I]$ and an assignment $s : VAR \longrightarrow U$ We write

$$s(\tfrac{a}{x})$$

to **denote** any assignment

$$s' : VAR \longrightarrow U$$

such that s, s' **agree** on all variables **except** on x and such that

$$s'(x) = a \quad \text{for certain} \quad a \in U$$

Classical Satisfaction

We introduce now a notion of a **satisfaction relation** $(\mathbf{M}, s) \models A$ that acts between structures, assignments and formulas of $\mathcal{L}$

It is the **satisfaction relation** that allows us to distinguish **one**

semantics for a given $\mathcal{L}$ from the **other**, and consequently one logic from the other

We define now only a classical satisfaction and the notion of classical predicate **tautology**

Classical Satisfaction

Definition

Given a predicate (first order) language $\mathcal{L} = \mathcal{L}(\mathbf{P}, \mathbf{F}, \mathbf{C})$

Let $\mathbf{M} = [U, I]$ be a structure for $\mathcal{L}$ and

$s : VAR \longrightarrow U$ be any assignment of $\mathcal{L}$ in $\mathbf{M}$

Let $A \in \mathcal{F}$ be any formula of $\mathcal{L}$

We define a **satisfaction relation**

$$(\mathbf{M}, s) \models A$$

that reads: " the assignment s **satisfies** the formula A in $\mathbf{M}$"

by **induction** on the complexity of A as follows

Classical Satisfaction

(i) A is **atomic formula**

$(\mathbf{M}, s) \models P(t_1, \ldots, t_n)$ if and only if $(s_I(t_1), \ldots, s_I(t_n)) \in P_I$

(ii) A **is not** atomic formula and has one of connectives of $\mathcal{L}$ as the main connective

$(\mathbf{M}, s) \models \neg A$ if and only if $(\mathbf{M}, s) \not\models A$

$(\mathbf{M}, s) \models (A \cap B)$ if and only if $(\mathbf{M}, s) \models A$ and $(\mathbf{M}, s) \models B$

$(\mathbf{M}, s) \models (A \cup B)$ if and only if $(\mathbf{M}, s) \models A$ or $(\mathbf{M}, s) \models B$ or both

$(\mathbf{M}, s) \models (A \Rightarrow B)$ if and only if ether $(\mathbf{M}, s) \not\models A$ or else $(\mathbf{M}, s) \models B$ or both

Classical Satisfaction

(iii) A **is not** atomic formula and A begins with one of the **quantifiers**

$(\mathbf{M}, s) \models \exists x A$ if and only if **there is** s' such that s, s' **agree** on all variables except on x, and

$$(\mathbf{M}, s') \models A$$

$(\mathbf{M}, s) \models \forall x A$ if and only if **for all** s' such that s, s' **agree** on all variables except on x, and

$$(\mathbf{M}, s') \models A$$

Classical Satisfaction

Observe that that the **truth** or **falsity** of $(\mathbf{M}, s) \models A$ depends only on the values of $s(x)$ for variables x which are actually **free** in the formula A.

This is why we often write the condition **(iii)** as follows

Classical Satisfaction

(iii)' $A(x)$ (with a free variable x) **is not** atomic formula and A begins with one of the **quantifiers**

$(\mathbf{M}, s) \models \exists x A(x)$ if and only if **there is** s' such that $s(y) = s'(y)$ such that for all $y \in VAR - \{x\}$,

$$(\mathbf{M}, s') \models A(x)$$

$(\mathbf{M}, s) \models \forall x A$ if and only if **for all** s' such that $s(y) = s'(y)$ for all $y \in VAR - \{x\}$,

$$(\mathbf{M}, s') \models A(x)$$

Satisfaction Relation Exercise

Exercise

For the structures $\mathbf{M}_i$, find assignments s_i, s'_i for $1 \leq i \leq 2$ such that

$$(\mathbf{M}_i, s_i) \models Q(x, c), \quad \text{and} \quad (\mathbf{M}_i, s'_i) \not\models Q(x, c)$$

where $Q \in \mathbf{P}$, $c \in \mathbf{C}$

The structures $\mathbf{M}_i$ are defined as follows (the interpretation I for each of them is specified only for symbols in the **atomic** formula $Q(x, c)$, and N denotes the set of natural numbers

$$\mathbf{M}_1 = [\{1\},\ Q_I := ,\ c_I : 1] \text{ and } \mathbf{M}_2 = [\{1, 2\},\ Q_I :\leq,\ c_I : 1]$$

Satisfaction Relation Exercise

Solution

Given Q(x,c). Consider

$$\mathbf{M}_1 = [\{1\},\ Q_I :=,\ c_I : 1]$$

Observe that **all** assignments

$$s : VAR \longrightarrow \{1\}$$

must be defined by a formula $s(x) = 1$ for all $x \in VAR$

We evaluate $s_I(x) = 1,\ s_I(c) = c_I = 1$

By definition

$$(\mathbf{M}_1, s) \models Q(x, c) \quad \text{if and only if} \quad (s_I(x),\ s_I(c)) \in Q_I$$

This means that $(1, 1) \in =$ what is **true** as $1 = 1$

We have proved

$$(\mathbf{M}_1, s) \models Q(x, c) \text{ for all assignments } s : VAR \longrightarrow \{1\}$$

Satisfaction Relation Exercise

Given Q(x,c). Consider

$$\mathbf{M}_2 = [\{1,2\},\ Q_I :\leq,\ c_I : 1]$$

Let $s : VAR \longrightarrow \{1,2\}$ be any assignment, such that

$$s(x) = 1$$

We evaluate $s_I(x) = 1,\ s_I(c) = 1$ and **verify** whether $(s_I(x),\ s_I(c)) \in Q_I$ i.e. whether $(1,1) \in \leq$

This is **true** as $1 \leq 1$

We have found s such that

$$(\mathbf{M}_2, s) \models Q(x,c)$$

In fact, have found uncountably many such assignments s

Satisfaction Relation Exercise

Given Q(x,c) and the structure

$$\mathbf{M}_2 = [\{1,2\},\ Q_I :\leq,\ c_I : 1]$$

Let now s' we be any assignment

$$s' : VAR \longrightarrow \{1,2\} \text{ such that } s'(x) = 2$$

We evaluate $s'_I(x) = 1, \quad s'_I(c) = 1$

We verify whether $s'_I(x),\ s'_I(c)) \in Q_I$, i.e. whether $(2,1) \in \leq$

This is **not true** as $2 \not\leq 1$

We have **found** $s' \neq s$ such that

$$(\mathbf{M}_2, s') \not\models Q(x,c)$$

In fact, have found uncountably many such assignments s'

Model Definition

Definition

Given a predicate language $\mathcal{L}$, a formula $A \in \mathcal{F}$, and a structure $\mathbf{M} = [U, I]$ for $\mathcal{L}$

$\mathbf{M}$ is a **model** for the formula A if and only if

$(\mathbf{M}, s) \models A$ for all $s : VAR \longrightarrow U$

We denote it as

$$\mathbf{M} \models A$$

For any set $\Gamma \subseteq \mathcal{F}$ of formulas of $\mathcal{L}$,

$\mathbf{M}$ is a **model** for Γ if and only if $\mathbf{M} \models A$ for all $A \in \Gamma$

We denote it as

$$\mathbf{M} \models \Gamma$$

Counter Model Definition

Definition

Given a predicate language $\mathcal{L}$, a formula $A \in \mathcal{F}$, and a structure $\mathbf{M} = [U, I]$ for $\mathcal{L}$

$\mathbf{M}$ is a **counter model** for the formula A if and only if **there is** an assignment $s : VAR \longrightarrow U$, such that $(\mathbf{M}, s) \not\models A$

We denote it as

$$\mathbf{M} \not\models A$$

Counter Model Definition

Definition

For any set $\Gamma \subseteq \mathcal{F}$ of formulas of $\mathcal{L}$,

$\mathbf{M}$ is a **counter model** for Γ if and only if

there is $A \in \Gamma$, such that $\mathbf{M} \not\models A$

We denote it as

$$\mathbf{M} \not\models \Gamma$$

Sentence Model

Observe that if a formula A is a **sentence** then the **truth** or **falsity** of satement

$$(\mathbf{M}, s) \models A$$

is completely independent of s

Hence if $(\mathbf{M}, s) \models A$ for some s, it holds for all s and the following holds

Fact

For any formula A of $\mathcal{L}$
If A is a **sentence**, then if there is an s such that

$$(\mathbf{M}, s) \models A$$

then **M** is a model fo A, i.e.

$$\mathbf{M} \models A$$

Formula Closure

We transform any formula A of $\mathcal{L}$ into a certain sentence by **binding** all its free variables. The resulting sentence is called a **closure** of A and is defined as follows

Definition

Given A of $\mathcal{L}$

By the **closure** of A we mean the formula obtained from A by prefixing in **universal** quantifiers all variables the arefree in A

If A **does not** have free variables, i.e. is a sentence, the **closure** of A is defined to be A itself

Obviously, a **closure** of any formula is always a sentence

Formula Closure Example

Example

Let A, B be formulas

$$(P(x_1, x_2) \Rightarrow \neg\exists x_2\, Q(x_1, x_2, x_3))$$

$$(\forall x_1 P(x_1, x_2) \Rightarrow \neg\exists x_2\, Q(x_1, x_2, x_3))$$

Their respective **closures** are

$$\forall x_1 \forall x_2 \forall x_3\, ((P(x_1, x_2) \Rightarrow \neg\exists x_2\, Q(x_1, x_2, x_3)))$$

$$\forall x_1 \forall x_2 \forall x_3\, ((\forall x_1 P(x_1, x_2) \Rightarrow \neg\exists x_2\, Q(x_1, x_2, x_3)))$$

Model, Counter Model Example

Example

Let $Q \in \mathbf{P}$, $\#Q = 2$ and $c \in \mathbf{C}$

Consider formulas

$$Q(x, c), \quad \exists x Q(x, c), \quad \forall x Q(x, c)$$

and the structures defined as follows.

$$\mathbf{M}_1 = [\{1\},\ Q_I := ,\ c_I : 1\ ] \quad \text{and} \quad \mathbf{M}_2 = [\{1, 2\},\ Q_I : \leq,\ c_I : 1\ ]$$

Directly from definition and above **Fact** we get that:

1. $\mathbf{M}_1 \models Q(x, c)$, $\mathbf{M}_1 \models \forall x Q(x, c)$, $\mathbf{M}_1 \models \exists x Q(x, c)$

2. $\mathbf{M}_2 \not\models Q(x, c)$, $\mathbf{M}_2 \not\models \forall x Q(x, c)$, $\mathbf{M}_2 \models \exists x Q(x, c)$

Example

Let $Q \in \mathbf{P}$, $\#Q = 2$ and $c \in \mathbf{C}$

Consider formulas

$$Q(x, c), \quad \exists x Q(x, c), \quad \forall x Q(x, c)$$

and the structures defined as follows.

$$\mathbf{M}_3 = [N, \ Q_I :\geq, \ c_I : 0], \quad \text{and} \quad \mathbf{M}_4 = [N, \ Q_I :\geq, \ c_I : 1]$$

Directly from definition and above **Fact** we get that:

3. $\mathbf{M}_3 \models Q(x, c), \quad \mathbf{M}_3 \models \forall x Q(x, c), \quad \mathbf{M}_3 \models \exists x Q(x, c)$

4. $\mathbf{M}_4 \not\models Q(x, c), \quad \mathbf{M}_4 \not\models \forall x Q(x, c), \quad \mathbf{M}_4 \models \exists x Q(x, c)$

True, False in **M**

Definition

Given a structure $\mathbf{M} = [U, I]$ for $\mathcal{L}$ and a formula A of $\mathcal{L}$

A is **true** in **M** and is written as

$$\mathbf{M} \models A$$

if and only if **all** assignments s of $\mathcal{L}$ in **M** **satisfy** A, i.e. when **M** is a **model** for A

A is **false** in **M** and written as

$$\mathbf{M} =\mid A$$

if and only if **there is no** assignment s of $\mathcal{L}$ in **M** that **satisfies** A

True, False in **M**

Here are some **properties** of the notions:

1. " A is **true** in **M**" written symbolically as

$$\mathbf{M} \models A$$

2. " A is **false** in **M**" written symbolically as

$$\mathbf{M} =\mid A$$

They are obvious under intuitive understanding of the notion of **satisfaction**

Their formal proofs are left as an exercise

Properties

Given a structure $\mathbf{M} = [U, I]$ and any formulas formula A, B of $\mathcal{L}$. The following properties hold

P1. A is **false** in **M** if and only if $\neg A$ is **true** in **M**, i.e.

$$\mathbf{M} \;=\!| \; A \quad \text{if and only if} \quad \mathbf{M} \models \neg A$$

P2. A is **true** in **M** if and only if $\neg A$ is **false** in **M**, i.e.

$$\mathbf{M} \models A \quad \text{if and only if} \quad \mathbf{M} \;=\!| \; \neg A$$

P3. It is **not** the case that **both** $\mathbf{M} \models A$ and $\mathbf{M} \models \neg A$, i.e. there **is no** formula A, such that

$$\mathbf{M} \models A \quad \text{and} \quad \mathbf{M} \;=\!| \; A$$

True, False in $\mathbf{M}$ Properties

Properties

P4. If $\mathbf{M} \models A$ and $\mathbf{M} \models (A \Rightarrow B)$, then $\mathbf{M} \models B$

P5. $(A \Rightarrow B)$ is **false** in $\mathbf{M}$ if and only if
$\mathbf{M} \models A$ and $\mathbf{M} \models \neg B$

$$\mathbf{M} =\!| \, (A \Rightarrow B) \text{ if and only if } \mathbf{M} \models A \text{ and } \mathbf{M} \models \neg B$$

P6. $\mathbf{M} \models A$ if and only if $\mathbf{M} \models \forall x A$

P7. A formula A is **true** in $\mathbf{M}$ if and only if its closure is **true** in $\mathbf{M}$

Valid, Tautology Definition

Definition

A formula A of $\mathcal{L}$ is a **predicate** tautology (is **valid**) if and only if $\mathbf{M} \models A$ for **all** structures $\mathbf{M} = [U, I]$

We also say

A formula A of $\mathcal{L}$ is a **predicate** tautology (is **valid**) if and only if A is **true** in **all** structures $\mathbf{M}$ for $\mathcal{L}$

We write

$$\models A \quad \text{or} \quad \models_p A$$

to denote that a formula A is **predicate** tautology (is **valid**)

Valid, Tautology Definition

We write

$$\models_p A$$

when there is a **need** to stress a distinction between **propositional** and **predicate** tautologies

otherwise we write

$$\models A$$

Predicate tautologies are also called **laws of quantifiers**.

Following the notation $\mathbf{T}$ we have established for the **set** of all propositional tautologies we denote by $\mathbf{T}_p$ the **set** of all predicate tautologies

We put

$$\mathbf{T}_p = \{A \text{ of } \mathcal{L} : \models_p A\}$$

Not a Tautology, Counter Model

Definition

For any formula A of predicate language $\mathcal{L}$

A **is not** a predicate tautology and denote it by

$$\not\models A$$

if and only if there is a structure $\mathbf{M} = [U, I]$ for $\mathcal{L}$, such that

$$\mathbf{M} \not\models A$$

We call such structure **M** a **counter-model** for A

Counter Model

In order to **prove** that a formula A **is not** a tautology one has to find a **counter-model** for A

It means one has to define the components of a structure $\mathbf{M} = [U, I]$ for $\mathcal{L}$, i.e.

a non-empty set U, called **universe** and

an interpretation I of $\mathcal{L}$ in the universe U

Moreover, one has to define an assignment $s : VAR \longrightarrow U$ and **prove** that that

$$(\mathbf{M}, s) \not\models A$$

Contradictions

We introduce now a notion of predicate **contradiction**

Definition

For any formula A of $\mathcal{L}$,

A is a **predicate contradiction** if and only if

A is **false** in **all** structures **M**

We denote it as $=\!|\ A$ and write symbolically

$$=\!|\ A \quad \text{if and only if} \quad \mathbf{M} =\!|\ A, \ \text{for } \mathbf{all} \text{ structures } \mathbf{M}$$

When there is a need to distinguish between propositional and predicate contradictions we also use symbol

$$=\!|_p\ A$$

Contradictions

Following the notation $\mathbf{C}$ for the set of all propositional **contradictions** we denote by $\mathbf{C}_p$ the set of all predicate contradictions, i.e.

$$\mathbf{C}_p = \{A \ \text{ of } \ \mathcal{L}(\mathbf{P}, \mathbf{F}, \mathbf{C}) : \ \ =|_p A\}$$

Directly from the contradiction definition we have the following duality property charecteristic for classical logic

Fact

For any formula A of a predicate language $\mathcal{L}$,

$$A \in \mathbf{T}_p \quad \text{if and only if} \quad \neg A \in \mathbf{C}_p$$

$$A \in \mathbf{C}_p \quad \text{if and only if} \quad \neg A \in \mathbf{T}_p$$

Proving Predicate TAutologies

We **prove**, as an example the following **basic** predicate tautology

Fact

For any formula $A(x)$ of $\mathcal{L}$,

$$\models \ (\forall x\, A(x) \Rightarrow \exists x\, A(x))$$

Proof

Assume that $\not\models (\forall x\, A(x) \Rightarrow \exists x\, A(x))$

It means that there is a structure

$$\mathbf{M} = [U, I] \text{ and } s : VAR \longrightarrow U, \text{ such that}$$

$$(\mathbf{M}, s) \not\models (\forall x\, A(x) \Rightarrow \exists x\, A(x))$$

Proving Predicate Tautologies

Observe that $(\mathbf{M}, s) \not\models (\forall x\, A(x) \Rightarrow \exists x\, A(x))$ is equivalent to

$$(\mathbf{M}, s) \not\models \forall x\, A(x) \text{ and } (\mathbf{M}, s) \not\models \exists x\, A(x)$$

By definition, $(\mathbf{M}, s) \not\models \forall x\, A(x)$ means that $(\mathbf{M}, s') \models A(x)$ for **all** s' such that s, s' agree on all variables except on x

At the same time $(\mathbf{M}, s) \not\models \exists x\, A(x)$ means that it is **not true** that **there is** s' such that s, s' agree on all variables except on x, and $(\mathbf{M}, s') \models A(x)$. This **contradiction** proves

$$\models (\forall x\, A(x) \Rightarrow \exists x\, A(x))$$

Disapproving Predicate Tautologies

We show now, as an example of a **counter model** construction that the converse implication to

$$\models \ (\forall x\, A(x) \Rightarrow \exists x\, A(x))$$

is not a predicate tautology i.e. the following holds

Fact

There is a formula A of $\mathcal{L}$, such that

$$\not\models \ (\exists x\, A(x) \Rightarrow \forall x\, A(x))$$

Proof

Observe that to prove the **Fact** we have to provide an example of an instance of a formula $A(x)$ and construct a **counter model** $\mathbf{M} = [U, I]$ for it

Proving Predicate Tautologies

Let $A(x)$ be an **atomic** formula

$$P(x, c) \quad \text{for any} \quad P \in \mathbf{P}, \quad \#P = 2$$

The needed instance is a formula

$$(\exists x\, P(x, c) \Rightarrow \forall x\, P(x, c))$$

We take as its **counter model** a structure

$$\mathbf{M} = [\, N, \quad P_I :<, \quad c_I : 3\,]$$

where N is set of natural numbers. We want to show

$$\mathbf{M} \not\models (\exists x\, P(x, c) \Rightarrow \forall x\, P(x, c))$$

It means we have to define an assignment s such that $s : VAR \longrightarrow N$ and

$$(\mathbf{M}, s) \not\models (\exists x\, P(x, c) \Rightarrow \forall x\, P(x, c))$$

Proving Predicate Tautologies

Let s be any assignment $s : VAR \longrightarrow N$

We show now

$$(\mathbf{M}, s) \models \exists x\, P(x, c)$$

Take any s' such that

$$s'(x) = 2 \quad \text{and} \quad s'(y) = s(y) \quad \text{for all } y \in VAR - \{x\}$$

We have $(2, 3) \in P_I$, as $2 < 3$

Hence we proved that **there exists** s' that agrees with s on all variables except on x and

$$(\mathbf{M}, s') \models P(x, c)$$

Proving Predicate Tautologies

But at the same time

$$(\mathbf{M}, s) \not\models \forall x\, P(x, c)$$

as for example for s' such that

$$s'(x) = 5 \quad \text{and} \quad s'(y) = s(y) \quad \text{for all} \quad y \in VAR - \{x\}$$

We have that $(2, 3) \notin P_I$, as $5 \not< 3$

This proves that the structure

$$\mathbf{M} = [N, P_I :<, \ c_I : 3\,]$$

is a **counter model** for $\forall x\, P(x, c)$

Hence we proved that

$$\not\models \ (\exists x\, A(x) \Rightarrow \forall x\, A(x))$$

Short Hand Solution of

$$\not\models \ (\exists x\ P(x,c) \Rightarrow \forall x\ P(x,c))$$

We take as its **counter model** a structure

$$\mathbf{M} = [\ N, \quad P_I :<, \quad c_I : 3\ ]$$

where N is set of natural numbers
The formula

$$(\exists x\ P(x,c) \Rightarrow \forall x\ P(x,c))$$

becomes in $\mathbf{M} = (N, P_I :<,\ c_I : 3)$ a mathematical statement (written with logical symbols):

$$\exists n\ n < 3 \ \Rightarrow\ \forall n\ n < 3$$

It is an obviously **false** statement in the set N of natural numbers, as there is $n \in N$, such that $n < 3$, for example $n = 2$, and it is **not true** that all natural numbers are smaller then 3

Chapter 8
Classical Predicate Semantics and Proof Systems

Slides Set 3

PART 3: Predicate Tautologies,
Equational Laws of Quantifiers

Predicate Tautologies

Predicate Tautologies

We have already proved the **basic** predicate tautology

$$\models \ (\forall x\, A(x) \Rightarrow \exists x\, A(x))$$

We **prove** now other three **basic** tautologies called

Dictum de Omni

For any formula $A(x)$ of $\mathcal{L}$,

$$\models \ (\forall x\, A(x) \Rightarrow A(t)), \qquad \models \ (\forall x\, A(x) \Rightarrow A(x))$$

$$\models \ (A(t) \Rightarrow \exists x\, A(x))$$

where t is a term, A(t) is a result of substitution of t for all free occurrences of x in $A(x)$, and t is **free for** x in A(x), i.e. **no** occurrence of a variable in t becomes a **bound** occurrence in A(t)

Proof of Dictum de Omni

Proof of

$$\models (\forall x\, A(x) \Rightarrow A(t)), \quad \models (\forall x\, A(x) \Rightarrow A(x))$$

is constructed in a sequence of the following steps

We leave details to complete as an exercise

S1

Consider a structure $\mathbf{M} = [U, I]$ and $s : VAR \longrightarrow U$

Let $t,\ u$ be two terms

Denote by t' a result of **replacing** in t all occurrences of a variable x by the term u, i.e.

$$t' = t(x/u)$$

Let s' results from s by **replacing** $s(x)$ by $s_I(u)$

We prove by induction over the length of t that

$$s_I(t(x/u)) = s_I(t') = s'_I(u)$$

S2

Let t be **free for** x in A(x)

$A(t)$ is a results from $A(x)$ by replacing t for all free occurrences of x in $A(x)$, i.e.

$$A(t) = A(x/t)$$

Let

$$s : VAR \longrightarrow U$$

and s' be obtained from s by replacing $s(x)$ by $s_I(u)$

We use

$$s_I(t(x/u)) = s_I(t') = s'_I(u)$$

and induction on the number of connectives and quantifiers in $A(x)$ and prove

$$(\mathbf{M}, s) \models A(x/t) \quad \text{if and only if} \quad (\mathbf{M}, s') \models A(x)$$

Proof of Dictum de Omni

S3

Directly from satisfaction definition and

$$(\mathbf{M}, s) \models A(x/t) \quad \text{if and only if} \quad (\mathbf{M}, s') \models A(x)$$

we get that for any $\mathbf{M} = [U, I]$ and any $s : VAR \longrightarrow U$,

$$\text{if } (\mathbf{M}, s) \models \forall x A(x), \text{ then } (\mathbf{M}, s) \models A(t)$$

This proves

$$\models (\forall x\, A(x) \Rightarrow A(t))$$

Observe that obviously a term x is **free for** x in A(x), so we also get as a **particular** case of $t = x$ that

$$\models (\forall x\, A(x) \Rightarrow A(x))$$

Dictum de Omni Restrictions

Proof of

$$\models \ (A(t) \Rightarrow \exists x\, A(x))$$

is included in detail in Section 3

Remark

The **restrictions** on terms in Dictum de Omni tautologies are **essential**

Here is a simple example explaining why they are needed in

$$\models \ (\forall x\, A(x) \Rightarrow A(t)), \qquad \models \ (\forall x\, A(x) \Rightarrow A(x))$$

Let $A(x)$ be a formula

$$\neg \forall y\, P(x, y) \quad \text{for} \quad P \in \mathbf{P}$$

Notice that a **term** $t = y$ is **not free for y** in A(x)

Dictum de Omni Restrictions

Consider the first formula in **Dictum de Omni** for $A(x) = \neg\forall y\, P(x, y)$ and term $t = y$

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

Take

$$\mathbf{M} = [N, I] \quad \text{for } \mathbf{I} \text{ such that } P_I := =$$

Obviously,

$$\mathbf{M} \models \forall x \neg\forall y\, P(x, y)$$

as

$$\forall m\, \neg\forall n (m = n)$$

is a **true** mathematical statement in the set N of natural numbers

Dictum de Omni Restrictions

$$\mathbf{M} \not\models \neg\forall y\; P(y, y)$$

as

$$\neg\forall n\; (n = n)$$

is a **false** statement for $n \in N$

The second **Dictum de Omni** formula is a particular case of the first

We have proved that without the restrictions on terms

$$\not\models\; (\forall x\, A(x) \Rightarrow A(t)) \quad \text{and} \quad \not\models\; (\forall x\, A(x) \Rightarrow A(x))$$

The example for $\models\; (A(t) \Rightarrow \exists x\, A(x))$ is similar

"t free for x in $A(x)$"

Here are some **useful** and easy to prove **properties** of the notion " term t **free for** x in $A(x)$ "

Properties

For any formula $A \in \mathcal{F}$ and any term $t \in \mathbf{T}$ the following properties hold

P1. Closed term t, i.e. term with no variables is free for any variable x in A

P2. Term t is free for any variable in A if none of the variables in t is bound in A

P3. Term $t = x$ is free for x in any formula A

P4. Any term is free for x in A if A contains no free occurrences of x

Predicate Tautologies

Here are some more **important** predicate **tautologies**

For any formulas $A(x), B(x), A, B$ of $\mathcal{L}$, where the formulas A, B **do not** contain any free occurrences of x the following holds

Generalization

$$\models ((B \Rightarrow A(x)) \Rightarrow (B \Rightarrow \forall x\, A(x)))$$

$$\models ((B(x) \Rightarrow A) \Rightarrow (\exists x B(x) \Rightarrow A))$$

Distributivity 1

$$\models (\forall x(A \Rightarrow B(x)) \Rightarrow (A \Rightarrow \forall x\, B(x)))$$

$$\models \forall x(A(x) \Rightarrow B) \Rightarrow (\exists x A(x) \Rightarrow B)$$

$$\models \exists x(A(x) \Rightarrow B) \Rightarrow (\forall x A(x) \Rightarrow B)$$

Restrictions

The **restrictions** that the formulas A, B **do not** contain any free occurrences of x is **essential** for both Generalization and Distributivity 1 tautologies

Here is a simple **example** explaining why they are needed

The relaxation of the **restrictions** would lead to the following disaster

Let A and B be both the same **atomic** formula P(x)

Thus x is **free** in A and we have the following instance of the first .Distributivity 1 tautology

$$.(\forall x(P(x) \Rightarrow P(x)) \Rightarrow (P(x) \Rightarrow \forall x\, P(x)))$$

Restrictions

Take

$$\mathbf{M} = [N, I] \quad \text{for } \mathbf{I} \text{ such that } P_I = ODD$$

where $ODD \subseteq N$ is the set of odd numbers

Let $s : VAR \longrightarrow N$

By definition of the interpretation i,

$$s_I(x) \in P_I \quad \text{if and only if} \quad s_I(x) \in ODD$$

Then obviously

$$(\mathbf{M}, s) \not\models \forall x\, P(x)$$

and $\mathbf{M} = [N, I]$ is a **counter model** for

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

as

$$\models \forall x(P(x) \Rightarrow P(x))$$

The examples for restrictions on other tautologies are similar.

Predicate Tautologies

Distributivity 2

For any formulas $A(x), B(x)$ of $\mathcal{L}$

$$\models \ (\exists x\ (A(x) \cap B(x)) \ \Rightarrow \ (\exists x A(x) \cap \exists x B(x)))$$

$$\models \ ((\forall x A(x) \cup \forall x B(x)) \Rightarrow \forall x\ (A(x) \cup B(x)))$$

$$\models (\forall x(A(x) \Rightarrow B(x)) \Rightarrow (\forall x A(x) \Rightarrow \forall x B(x)))$$

The converse implications to the above **are not** predicate tautologies

The **counter models** are provided in the Section 3

De Morgan Laws

For any formulas $A(x), B(x)$ of $\mathcal{L}$,

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

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

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

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

We prove the first law as an example

The proofs of all other laws are similar

De Morgan Laws

Proof of

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

We carry the proof by **contradiction**

Assume that

$$\not\models \models (\neg \forall x A(x) \Rightarrow \exists x \neg A(x))$$

By definition, there is

$$\mathbf{M} = [U, I] \quad \text{and} \quad s : VAR \longrightarrow U$$

such that

$$(\mathbf{M}, s) \models \neg \forall x A(x)) \quad \text{and} \quad (\mathbf{M}, s) \not\models \exists x \neg A(x)$$

De Morgan Laws

Consider

$$(\mathbf{M}, s) \models \neg\forall x A(x)$$

By satisfaction definition

$$(\mathbf{M}, s) \not\models \forall x A(x)$$

This holds only if for **all** s', such that s, s' agree on all variables except on x,

$$(\mathbf{M}, s') \not\models A(x)$$

De Morgan Laws

Consider now

$$(\mathbf{M}, s) \not\models \exists x \neg A(x)$$

This holds only if **there is no** s', such that

$$(\mathbf{M}, s') \models \neg A(x)$$

i.e. there **is no** s', such that $(\mathbf{M}, s') \not\models A(x)$

This means that **for all** s',

$$(\mathbf{M}, s') \models A(x)$$

Contradiction with already proved

$$(\mathbf{M}, s') \not\models A(x)$$

This **ends** the proof

Quantifiers Alternations

For any formula $A(x, y)$ of $\mathcal{L}$,

$$\models (\exists x \forall y A(x, y) \Rightarrow \forall y \exists x A(x, y))$$

The converse implication

$$(\forall y \exists x A(x, y) \Rightarrow \exists x \forall y A(x, y))$$

is not a predicate **tautology**

Here is a proof

Take as $A(x, y)$ an atomic formula $R(x, y)$

Consider the **instance** formula

$$(\forall y \exists x R(x, y) \Rightarrow \exists x \forall y R(x, y))$$

Quantifiers Alternations

We construct now a counter model for the instance formula

$$(\forall y \exists x R(x, y) \Rightarrow \exists x \forall y R(x, y))$$

Take a structure

$$\mathbf{M} = [R, I]$$

where R is the set of real numbers and $R_I :<$

The instance formula becomes a mathematical statement

$$(\forall y \exists x (x < y) \Rightarrow \exists x \forall y (x < y))$$

that obviously **false** in the set of real numbers

We proved

$$\not\models (\forall y \exists x A(x, y) \Rightarrow \exists x \forall y A(x, y))$$

Equational Laws of Quantifiers

Logical Equivalence

The most frequently used **laws of quantifiers** have a form of a logical equivalence, symbolically written as $\equiv$

Logical equivalence $\equiv$ **is not** a new logical connective but is just a very useful **symbol**

Logical equivalence $\equiv$ has the same properties as the mathematical equality $=$ and can be used in a similar way as we use the equality

Note that we use the same equivalence symbol $\equiv$ and the tautology symbol $\models$ for **propositional** and **predicate** languages when there is no confusion

Logical Equivalence

We define formally the **logical equivalence** $\equiv$ as follows.

Definition of Logical Equivalence

For any formulas A, B of the **predicate** language $\mathcal{L}$,

$$A \equiv B \text{ if and only if } \models (A \Rightarrow B) \text{ and } \models (B \Rightarrow A)$$

Remark that the predicate language $\mathcal{L}$ we defined the **semantics** for does not include the equivalence connective $\Leftrightarrow$. If it does we **extend** the satisfaction definition in a natural way and adopt the following, natural definition

Definition

For any formulas $A, B \in \mathcal{F}$ of the **predicate language** $\mathcal{L}$ with the equivalence connective $\Leftrightarrow$

$$A \equiv B \quad \text{if and only if} \quad \models (A \Leftrightarrow B)$$

Logical Equivalence Theorems

The **basic** theorems establishing relationship between propositional and some predicate **tautologies** are as follows

Tautologies Theorem
If a formula A is a propositional tautology, then by **substituting** for propositional variables in A any formula of the **predicate** language $\mathcal{L}$ we obtain a formula which is a predicate tautology

Equivalences Theorem

Given propositional formulas A, B

If $A \equiv B$ is a propositional **equivalence**, and A', B' are formulas of the predicate language L obtained by a **substitution** of any formulas of $\mathcal{L}$ for propositional variables in A and B, respectively,

then

$$A' \equiv B'$$

holds under predicate semantics

Logical Equivalence Example

Example

Consider the following propositional logical equivalence

$$(a \Rightarrow b) \equiv (\neg a \cup b)$$

Substituting

$$\exists x P(x, z) \text{ for } a \quad \text{and} \quad \forall y R(y, z) \text{ for } b$$

we get by the **EquivalencesTheorem** that the following logical equivalence holds

$$(\exists x P(x, z) \Rightarrow \forall y R(y, z)) \equiv (\neg \exists x P(x, z) \cup \forall y R(y, z))$$

Equivalence Substitution

We prove in similar way as in the propositional case the following.

Equivalence Substitution Theorem

Let a formula B_1 be obtained from a formula A_1 by a **substitution** of a formula B for **one** or **more** occurrences of a sub-formula A of A_1, what we denote as

$$B_1 = A_1(A/B)$$

Then the following holds for any formulas A, A_1, B, B_1 of $\mathcal{L}$

$$\text{If} \quad A \equiv B, \quad \text{then} \quad A_1 \equiv B_1$$

Logical Equivalence Theorem

Directly from the Dictum de Omi and theGeneralization tautologies we get the proof of the following theorem useful for building new logical equivalences from the old, already known ones

E- Theorem

For any formulas $A(x), B(x)$ of $\mathcal{L}$

$$\text{if } A(x) \equiv B(x), \text{ then } \forall x A(x) \equiv \forall x B(x)$$

$$\text{if } A(x) \equiv B(x), \text{ then } \exists x A(x) \equiv \exists x B(x)$$

Logical Equivalence Example

Example

We know from the previous example that

$$(\exists x P(x,z) \Rightarrow \forall y R(y,z)) \equiv (\neg \exists x P(x,z) \cup \forall y R(y,z))$$

We get, as the direct consequence of the above theorem the following logical equivalence

$$\forall z(\exists x P(x,z) \Rightarrow \forall y R(y,z)) \equiv \forall z(\neg \exists x P(x,z) \cup \forall y R(y,z))$$

$$\exists z(\exists x P(x,z) \Rightarrow \forall y R(y,z)) \equiv \exists z(\neg \exists x P(x,z) \cup \forall y R(y,z))$$

Equational Laws of Quantifiers

We concentrate now only on these laws of quantifiers which have a form of a logical equivalence

They are called the **equational laws** of quantifiers

Directly from the logical equivalence definition and the De Morgan tautologies we get the following laws

Equational Laws of Quantifiers

De Morgan Laws

For any formulas $A(x),\ B(x)$ of $\mathcal{L}$

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

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

We now apply them to show that the **quantifiers** can be defined one by the other i.e. that the following Definability Laws hold

Equational Laws of Quantifiers

Definability Laws

For any formula $A(x)$ of $\mathcal{L}$

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

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

The first law is often used as a **definition** of the universal quantifier in terms of the existential one (and negation)

The second law is a **definition** of the existential quantifier in terms of the universal one (and negation)

Proof of

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

Substituting any formula $A(x)$ for a variable a in the propositional equivalence $a \equiv \neg\neg a$
we get by the **Equivalence Theorem** that

$$A(x) \equiv \neg\neg A(x)$$

Applying the **E-Theorem** to the above we obtain

$$\exists x A(x) \equiv \exists x \neg\neg A(x)$$

By the **De Morgan Law**

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

By the **Equivalence Substitution Theorem**

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

This ends the proof

Proof of

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

Substituting any formula $A(x)$ for a variable a in the propositional equivalence $a \equiv \neg\neg a$
we get by the **Equivalence Theorem** that

$$A(x) \equiv \neg\neg A(x)$$

Applying the **E-Theorem** to the above we obtain

$$\forall x A(x) \equiv \forall x \neg\neg A(x)$$

By the **De Morgan Law** and **Equivalence Substitution Theorem**

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

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

This ends the proof

Equational Laws of Quantifiers

Other important equational laws are the following **introduction** and **elimination** laws

Listed equivalences are **not independent**, some of them are the consequences of the others

Introduction and Elimination Laws

If B is a formula such that B **does not** contain any free occurrence of x, then the following logical equivalences hold for any formula $A(x)$ of $\mathcal{L}$

$$\forall x(A(x) \cup B) \equiv (\forall x A(x) \cup B)$$

$$\forall x(A(x) \cap B) \equiv (\forall x A(x) \cap B)$$

$$\exists x(A(x) \cup B) \equiv (\exists x A(x) \cup B)$$

$$\exists x(A(x) \cap B) \equiv (\exists x A(x) \cap B)$$

Equational Laws of Quantifiers

Introduction and Elimination Laws

$$\forall x(A(x) \Rightarrow B) \equiv (\exists x A(x) \Rightarrow B)$$

$$\exists x(A(x) \Rightarrow B) \equiv (\forall x A(x) \Rightarrow B)$$

$$\forall x(B \Rightarrow A(x)) \equiv (B \Rightarrow \forall x A(x))$$

$$\exists x(B \Rightarrow A(x)) \equiv (B \Rightarrow \exists x A(x))$$

As we said before, the equivalences **are not** independent

We show now as an **example** the proof of the third one from the first two

Equational Laws of Quantifiers

We write this proof in a short, symbolic way as follows

$$
\begin{aligned}
\exists x(A(x) \cup B) &\overset{\text{law}}{\equiv} \neg\forall x\neg(A(x) \cup B) \\
&\overset{\text{thms}}{\equiv} \neg\forall x(\neg A(x) \cap \neg B) \\
&\overset{\text{law}}{\equiv} \neg(\forall x\neg A(x) \cap \neg B) \\
&\overset{\text{law,thm}}{\equiv} (\neg\forall x\neg A(x) \cup \neg\neg B) \\
&\overset{\text{thm}}{\equiv} (\exists x A(x) \cup B)
\end{aligned}
$$

We leave completion and explanation of all details as it as and **exercise**

Equational Laws of Quantifiers

Distributivity Laws

Let $A(x), B(x)$ be any formulas with a free variable x

Law of distributivity of universal quantifier over conjunction

$$\forall x\,(A(x) \cap B(x)) \equiv (\forall x A(x) \cap \forall x B(x))$$

Law of distributivity of existential quantifier over disjunction

$$\exists x\,(A(x) \cup B(x)) \equiv (\exists x A(x) \cup \exists x B(x))$$

Equational Laws of Quantifiers

Alternations of Quantifiers

Let $A(x, y)$ be any formula with a free variables x, y

$$\forall x \forall y\ (A(x, y)) \equiv \forall y \forall x\ (A(x, y))$$

$$\exists x \exists y\ (A(x, y)) \equiv \exists y \exists x\ (A(x, y))$$

Equational Laws of Quantifiers

Renaming the Variables

Let $A(x)$ be any formula with a free variablex and let y be a variable that **does not occur** in A(x) y, then the following holds

$$\forall x A(x) \equiv \forall y A(y)$$

$$\exists x A(x) \equiv \exists y A(y)$$

Equational Laws of Quantifiers

Restricted De Morgan Laws

For any formulas $A(x), B(x)$ of $\mathcal{L}$

$$\neg\forall_{B(x)}\, A(x) \equiv \exists_{B(x)}\, \neg A(x)$$

$$\neg\exists_{B(x)}\, A(x) \equiv \forall_{B(x)} \neg A(x)$$

Equational Laws of Quantifiers

Here is a poof of first equality

The proof of the second one is similar and is left as an exercise.

$$\neg\forall_{B(x)} A(x) \equiv (\neg\forall x\ (B(x) \Rightarrow A(x)) \equiv$$

$$\neg\forall x\ (\neg B(x) \cup A(x)) \equiv \exists x\ \neg(\neg B(x) \cup A(x)) \equiv$$

$$\exists x\ (\neg\neg B(x) \cap \neg A(x)) \equiv \exists x\ (B(x) \cap \neg A(x)) \equiv \exists_{B(x)} \neg A(x))$$

Restricted Introduction and Elimination Laws

Let B be a formula that **does not** contain any free occurrence of x

then the following logical equivalences hold for any formulas $A(x), B(x), C(x)$ of $\mathcal{L}$

$$\forall_{C(x)}(A(x) \cup B) \equiv (\forall_{C(x)}A(x) \cup B)$$

$$\exists_{C(x)}\,(A(x) \cap B) \equiv (\exists_{C(x)}\,A(x) \cap B)$$

$$\forall_{C(x)}(A(x) \Rightarrow B) \equiv (\exists_{C(x)}A(x) \Rightarrow B)$$

$$\forall_{C(x)}(B \Rightarrow A(x)) \equiv (B \Rightarrow \forall_{C(x)}A(x))$$

The **proofs** are similar to the proof of the restricted De Morgan Laws. The similar generalization of the other Introduction and Elimination Laws for restricted domain quantifiers fails

Equational Laws of Quantifiers

We prove by constructing proper counter-models the following.

$$\exists_{C(x)}(A(x) \cup B) \not\equiv (\exists_{C(x)} A(x) \cup B)$$

$$\forall_{C(x)}(A(x) \cap B) \not\equiv (\forall_{C(x)} A(x) \cap B)$$

$$\exists_{C(x)}(A(x) \Rightarrow B) \not\equiv (\forall_{C(x)} A(x) \Rightarrow B)$$

$$\exists_{C(x)}(B \Rightarrow A(x)) \not\equiv (B \Rightarrow \exists x A(x))$$

Equational Laws of Quantifiers

Nevertheless it is possible to correctly generalize them all as to cover quantifiers with **restricted domain**

We show now how we get the correct generalization of

$$\exists_{C(x)}(A(x) \cup B) \not\equiv (\exists_{C(x)} A(x) \cup B)$$

We leave the other cases an **exercise**

Equational Laws of Quantifiers

Example

The correct restricted quantifiers equality is

$$\exists_{C(x)}(A(x) \cup B) \equiv (\exists_{C(x)}A(x) \cup (\exists x\, C(x) \cap B))$$

We derive it as follows.

$$\exists_{C(x)}(A(x) \cup B) \equiv \exists x(C(x) \cap (A(x) \cup B)) \equiv$$

$$\exists x((C(x) \cap A(x)) \cup (C(x) \cap B)) \equiv (\exists x(C(x) \cap A(x)) \cup \exists x(C(x) \cap B))$$

$$\equiv \exists_{C(x)}A(x) \cup (\exists x\, C(x) \cap B))$$

We leave it as an exercise to specify and write references to transformation or equational laws used at each step of the computation

Chapter 8
Classical Predicate Semantics and Proof Systems

Slides Set 3

PART 4: Proof Systems: Soundness and Completeness

Proof Systems: Soundness and Completeness

We adopt now general definitions from chapter 4 concerning **proof systems** to the case of classical first order (predicate) logic

Chapters 4 and 5 contain a great array of examples, exercises, homework problems explaining in a great detail all notions we introduce here for the predicate case

The **examples** and **exercises** we provide here are not numerous and are restricted to the laws of quantifiers

Proof Systems

Given a predicate language

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

Any **proof system**

$$S = (\mathcal{L},\ \mathcal{F},\ LA,\ \mathcal{R})$$

is a **predicate** (first order) proof system

The predicate proof system S is a **Hilbert** proof system if the set $\mathcal{R}$ of its rules contains the Modus Ponens rule

$$(MP)\ \frac{A\ ;\ (A \Rightarrow B)}{B}$$

where $A, B \in \mathcal{F}$

Proof Systems

Semantic Link: Logical Axioms LA

We want the set LA of logical axioms to be a non-empty set of classical predicate tautologies, i.e.

$$LA \subseteq \mathbf{T}_p$$

where

$$\mathbf{T}_p = \{A \text{ of } \mathcal{L}_{\{\neg, \cap, \cup, \Rightarrow, \neg\}}(\mathbf{P}, \mathbf{F}, \mathbf{C}) : \models_p A\}$$

We use symbols

$$\models_p, \ \mathbf{T}_p$$

to stress the fact that we talk about predicate language and classical predicate tautologies

Rules of Inference

Semantic Link 2: Rules of Inference $\mathcal{R}$

We want the the **rules** of inference $r \in \mathcal{R}$ of S to preserve truthfulness. Rules that do so are called **sound**

Definition

Given an inference rule $r \in \mathcal{R}$ of the form

$$(r) \quad \frac{P_1 \ ; \ P_2 \ ; \ .... \ ; \ P_m}{C}$$

where $P_1.P_2, \ldots, P_m, C \in \mathcal{F}$

We say that the rule (r) is **sound** if and only if the following condition holds for **all** structures $\mathbf{M} = [U, I]$ for $\mathcal{L}$

$$\text{If } \mathbf{M} \models \{P_1, P_2, .P_m\} \text{ then } \mathbf{M} \models C$$

Exercise

Prove the soundness of the rule

$$(r) \quad \frac{\forall x A(x)}{\exists x A(x)}$$

Proof

Assume that (r) is **not sound**

It means that **there is** a structure $\mathbf{M} = [U, I]$, such that

$$\mathbf{M} \models \forall x A(x) \quad \text{and} \quad \mathbf{M} \not\models \exists x\, A(x)$$

Let $(\mathbf{M}, s) \models \forall x\, A(x)$ and $(\mathbf{M}, s) \not\models \exists x\, A(x)$

It means that $(\mathbf{M}, s') \models A(x)$ for all s' such that s, s' agree on all variables except on x, and it is **not true** that there is s' such that s, s' agree on all variables except on x , and $(\mathbf{M}, s') \models A(x)$

This is impossible and this **contradiction** proves soundness of (r)

Exercise

Prove that the rule

$$(r) \quad \frac{\exists x A(x)}{\forall x A(x)}$$

is **not sound**

Proof

Observe that to prove that the rule (r) is **not sound** we have to provide an example of an instance of a formula $A(x)$ and construct a counter model

Let $A(x)$ be an atomic formula P(x,c), for any $P \in \mathbf{P}$, $\#P = 2$

We take as a counter model a structure

$$\mathbf{M} = (N, \quad P_I :<, \quad c_I : 3)$$

where N is the set of natural numbers

Rules of Inference

Here is a "shorthand" solution

The atomic formula $(\exists x\ P(x, c)$ becomes in

$$\mathbf{M} = (N, \quad P_I :<, \quad c_I : 3)$$

a **true** mathematical statement (written with logical symbols):

$$\exists n\ n < 3$$

The formula $(\forall x\ P(x, c)$ becomes a mathematical statement

$$\forall n\ n < 3$$

which is an obviously **false** in the set N of natural numbers

This proves that the the rule (r) is **not sound**

Rules of Inference

Definition of Strongly Sound Rule

An inference rule $r \in \mathcal{R}$ of the form

$$(r) \qquad \frac{P_1 \;\; ; \;\; P_2 \;\; ; \;\; .... \;\; ; \;\; P_m}{C}$$

is **strongly sound** if the following condition holds for all structures $\mathbf{M} = [U, I]$ for $\mathcal{L}$

$$\mathbf{M} \models \{P_1, P_2, .P_m\} \quad \text{if and only if} \quad \mathbf{M} \models C$$

We can, and we do state it informally as

(r) is **strongly sound** if and only if $P_1 \cap P_2 \cap \; \ldots \; \cap P_m \equiv C$

Rules of Inference

Example

The sound rule

$$(r1) \quad \frac{\neg\forall x A(x)}{\exists x \neg A(x)}$$

is **strongly sound** by De Morgan Laws

Example

The sound rule

$$(r2) \quad \frac{\forall x A(x)}{\exists x A(x)}$$

is **not strongly sound** by exercise above

Soundness

Definition of Sound Proof System

Given the **predicate** (first order) proof system

$$S = (\mathcal{L},\ \mathcal{F},\ LA,\ \mathcal{R})$$

We say that S is **sound** if the following conditions hold

(1) $LA \subseteq \mathbf{T}_p$

(2) Each rule of inference $r \in \mathcal{R}$ is **sound**

The proof system S is **strongly sound** if the condition **(2)** is replaced by the following condition **(2')**

(2') Each rule of inference $r \in \mathcal{R}$ is **strongly sound**

Soundness Theorem

When we define (develop) a proof system S our first goal is to make sure that it is a "sound" one

It means that that all we **prove** in it is true. The following theorem establishes this goal

Soundness Theorem for S

Given a predicate proof system S

For any $A \in \mathcal{F}$, the following implication holds.

$$\text{If } \vdash_S A \quad \text{then} \quad \models_p A$$

We write it in a more concise form as

$$\mathbf{P}_S \subseteq \mathbf{T}_p$$

Soundness Theorem

Proof of Soundness Theorem

Observe that if we have already proven that S is **sound** as stated in the definition the proof of the implication

$$\text{If } \vdash_S A \quad \text{then} \quad \models_p A$$

is a straightforward application of the mathematical induction over the length of the formal proof of the formula A

It means that in order to prove the Soundness Theorem for a proof system S it is enough to **verify** the two conditions of the soundness definition, i.e. to verify

(1) $LA \subseteq \mathbf{T}_p$ and

(2) each rule of inference $r \in \mathcal{R}$ is **sound**

CompletessTheorem

Proving **Soundness Theorem** for any proof system S is indispensable and moreover, the proof is quite easy

The next step in developing a **logic** (classical predicate logic in our case now) is to answer the following necessary and difficult question

Given a proof system S about which we know that all it **proves** is true (**tautology**)

Can we **prove** *all we know to be true ?*. It means:

Can S prove all **tautologies***?*

Proving the following **theorem** establishes this goal

CompletenessTheorem

Completeness Theorem for S

Given a predicate proof system S

For any $A \in \mathcal{F}$, the following holds

$$\vdash_S A \quad \text{if and only if} \quad \models_p A$$

We write it in a more concise form as

$$\mathbf{P}_S = \mathbf{T}_p$$

CompletenessTheorem

The **Completeness Theorem** consists of two parts

Part 1: **Soundness Theorem**

$$\mathbf{P}_S \subseteq \mathbf{T}_p$$

Part 2: **Completeness part** of the **Completeness Theorem**

$$\mathbf{T}_p \subseteq \mathbf{P}_S$$

CompletenessTheorem

There are many methods and techniques fo r**proving** the CompletenessTheorem

It applies even for classical proof systems (logics) alone

Non-classical logics often require **new** and usually very sophisticated **methods**

CompletenessTheorem

We presented two very different **proofs** of the **Completeness Theorem** for classical propositional Hilbert style proof system in chapter 5

Then we presented yet another very different **constructive** proofs for automated theorem proving systems for classical propositional logic chapter 6

As a next step we present a standard proof of the **Completeness Theorem** for Hilbert style proof system for classical predicate logic in the next **chapter 9**