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

Anita Wasilewska

Chapter 2
Introduction to Classical Logic Languages and Semantics

CHAPTER 2 SLIDES

Chapter 2
Introduction to Classical Logic Languages and Semantics

Slides Set 1

PART 1: Classical Logic Model

PART 2: Propositional Language

PART 3: Propositional Semantics

PART 4: Examples of Propositional Tautologies

Slides Set 2

PART 5: Predicate Language

PART 6: Predicate Tautologies - Laws for Quantifiers

Chapter 2

Introduction to Classical Logic Languages and Semantics

Slides Set 1

PART 1: Classical Logic Model

Very Short History

Logic Origins:

Stoic school of philosophy (3rd century B.C.)

Tthe most eminent representative was Chryssipus

Modern Origins:

Mid-19th century

English mathematician G. Boole, who is sometimes regarded as the founder of mathematical logic

First Axiomatic System:

In 1879 by German logician G. Frege.

Logic

Logic builds symbolic **models** of our world

Logic builds the **models** in order to describe **formally** the ways we reason in and about our world

Logic also poses questions about **correctness** of such **models** and develops tools to answer them

Classical Model Assumptions

Assumption 1

Classical logic **model** admits only two logical values

Why two logical values only?

Classical logic was created to model the **reasoning principles** of mathematics

We expect from **mathematical theorems** to be always either true or false and the **reasoning** leading to them should guarantee this without any ambiguity

Classical Model Assumptions

Assumption 2

1. The language in which we **reason** uses **sentences**

2. The **sentences** are build up from basic assertions about the world using special **words** or **phrases**:

”not”, ”not true” , ”and”, ”or”, ” implies”, ”if ..... then”, ”from the fact that .... we can deduce”, ” if and only if”, ”equivalent”, ”every”, ”for all”, any”, ”some”, ” exists”

3. We use **symbols** do denote **basic assertions** and special **words** or **phrases**

Hence the name **symbolic logic**

Logic

Logic studies the **behavior** of the special words and phrases

Special **words** and **phrases** have accepted intuitive meanings

Logic builds **models** to formalize these **intuitive meanings**

To do so we first **define** formal **symbolic languages** and then define a formal meaning of their symbols

The formal meaning is called **semantics**

Propositional Connectives

The **symbols** for he special words and phrases are called **propositional connectives**

There are different choices of **symbols** for the propositional connectives; we adopt the following:

$\neg$ for ”not”, ”not true”

$\cap$ for ”and”

$\cup$ for ör”

$\Rightarrow$ for ” implies” , ”if ..... then”, ”from the fact that... we can deduce”

$\Leftrightarrow$ for ” if and only if”, ”equivalent”

The **names** for the **propositional connectives** are:

negation for $\neg$

conjunction, for $\cap$, **disjunction** for $\cup$

implication for $\Rightarrow$, and **equivalence** for $\Leftrightarrow$

Propositional Logic

Restricting our attention to the role of **propositional connectives** yields to what is called **propositional logic**

The basic components of the **propositional logic** are a propositional language and a propositional semantics

The **propositional logic** is a quite simple model to **justify**, **describe** and **develop**

We devote first few chapters to it. We do it both for its own sake and because it provides a good background for developing and understanding more difficult languages and logics to follow

Quantifiers and Predicate Logic

Quantifiers

We use symbols:

$\forall$ for "every", "any", "all"

$\exists$ for "some" ," exists", "there is"

The symbols $\forall$, $\exists$ are called **quantifiers**

Consideration and study of the **role** of propositional connectives and quantifiers leads to what is called a **predicate logic**

Quantifiers and Predicate Logic

The basic components of the **predicate logic** are predicate language and predicate semantics

The **predicate logic** is a much more complicated model

We **develop** and **study** it in full formality in chapters following this introduction and examination of the **propositional logic** model

Chapter 2
Introduction to Classical Logic Languages and Semantics

Slides Set 1

PART 2: Propositional Language

Propositional language is a quite simple, symbolic language into which we can **translate (represent)** sentences of a natural language

Example

Consider natural language sentence
" If $2 + 2 = 5$, then $2 + 2 = 4$"

We translate it into the **propositional language** as follows

We **denote** the **basic assertion** (proposition) "$2 + 2 = 5$" by a variable, let's say a, and the proposition "$2 + 2 = 4$" by a variable b

We write a connective $\Rightarrow$ for "if ..... then"

As a result we obtain a propositional language **formula**

$$(a \Rightarrow b)$$

Propositional Translation

Exercise

Translate a natural language sentence **S**

"The fact that it is not true that at the same time 2+2 = 4 and 2+2 = 5 implies that 2+2 = 4"

into a corresponding propositional language **formula**

We carry the translation as follows

1. We **identify** all **words** and **phrases** representing the **logical connectives** and we re-write the sentence **S** in a simpler form introducing parenthesis to better express its meaning

Propositional Translation

The sentence **S** becomes:

" If not $(2+2=4$ and $2+2=5)$ then $2+2=4$"

2.

We identify the **basic assertions** (propositions) and **assign** propositional variables to them:

$$a: \ "2+2=4" \quad \text{and} \qquad b: \ "2+2=5"$$

Step 3

We write the propositional language **formula**:

$$(\neg(a \cap b) \Rightarrow a)$$

Syntax

A formal description of **symbols** and the definition of the set of **formulas** is called a **syntax** of a **symbolic** language

We use the word syntax to stress that the **formulas** do not carry neither formal meaning nor a logical value
We **assign** the meaning and logical value to syntactically defined **formulas** in a **separate** step
This next, separate step is called a **semantics** of the given symbolic language

A given **symbolic** language can have different semantics and the different semantics can define **different logics**

Natural Languages

One can think about a **natural language** as a set $\mathcal{W}$ of all words and sentences based on a given alphabet $\mathcal{A}$

This leads to a simple, abstract **model** of a **natural language** NL as a pair

$$NL = (\mathcal{A}, \mathcal{W})$$

Some natural languages share the same alphabet, some have different alphabets

All of them face serious **problems** with a proper recognition and definitions of accepted **words** and complex **sentences**

Symbolic Languages

We do not want the symbolic languages to share the difficulties of the natural languages

We **define** their components **precisely** and in such a way that their recognition and correctness will be easily **decided**
We call their words and sentences formulas and denote the set of all **formulas** by $\mathcal{F}$

We **define** a **symbolic language** as a pair

$$SL = (\mathcal{A}, \mathcal{F})$$

Symbolic Languages Categories

We distinguish two categories of symbolic languages:

propositional and **predicate**

We define first the propositional language

The definition of the predicate language, with its much more complicated **structure** will follow

Propositional Language Definition

Definition

By a **propositional language** $\mathcal{L}$ we understand a pair

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

where $\mathcal{A}$ is called propositional **alphabet**

$\mathcal{F}$ is called a set of all well formed **formulas**

Language Components: Alphabet

1. Alphabet $\mathcal{A}$

The alphabet $\mathcal{A}$ consists of

a countably infinite set VAR of **propositional variables**,

a finite set of **propositional connectives**, and

a set of two **parenthesis**

We denote the propositional variables by letters

$$a,\ b,\ c,\ p,\ q,\ r,\ ....$$

with indices if necessary. It means that we can also use

$$a_1, a_2, ..., b_1, b_2, ...$$

as symbols for propositional variables

Language Components: Alphabet

Propositional connectives are:

$$\neg, \quad \cap, \quad \cup, \quad \Rightarrow, \quad \Leftrightarrow$$

The connectives have well established names

The connectives names are:

negation, conjunction, disjunction, implication, and **equivalence (biconditional)**

for the connectives $\neg$, $\cap$, $\cup$, $\Rightarrow$, and $\Leftrightarrow$, respectively

Parenthesis are symbols (and)

Language Components: Formulas

Formulas are expressions build by means of elements of the alphabet $\mathcal{A}$. We denote formulas by capital letters $A,\ B,\ C,\ D, ...$, with indices, if necessary.
The set $\mathcal{F}$ of **all formulas** of the propositional language $\mathcal{L}$ is **defined recursively** as follows

1. Base step: all propositional variables are are **formulas**

They are called **atomic formulas**

2. Recursive step: for any already defined **formulas** A, B, the expressions

$$\neg A,\ (A \cap B),\ (A \cup B),\ (A \Rightarrow B),\ (A \Leftrightarrow B)$$

are also **formulas**

3. Only those expressions are **formulas** that are determined to be so by means of conditions **1.** and **2.**

Formulas Example

By the definition, any propositional variable is a **formula**. Let's take two variables a and b.

By the **recursive step** we get that

$$(a \cap b),\ (a \cup b),\ (a \Rightarrow b),\ (a \Leftrightarrow b),\ \neg a,\ \neg b$$

are **formulas**

The **recursive step** applied again produces for example

some **formulas** :

$$\neg(a \cap b),\ \ ((a \Leftrightarrow b) \cup \neg b),\ \ \neg\neg a,\ \ \neg\neg(a \cap b)$$

Formulas

Observe that we listed only few formulas obtained in the first recursive step

As as the recursive process **continue** we obtain a set of well formed of **formulas**

The set of all formulas is countably infinite

Formulas

Remark that we put parenthesis within the **formulas** in a way to avoid ambiguity

The expression

$$a \cap b \cup a$$

is ambiguous

We don't know whether it represents a formula

$$(a \cap b) \cup a \quad \text{or a formula} \quad a \cap (b \cup a)$$

Observe that **neither** of formulas $a \cap b \cup a$, $(a \cap b) \cup a$ or $a \cap (b \cup a)$ is a **well formed formula**

Exercises

Exercise

Consider a following set

$$\mathcal{S} = \{\neg a \Rightarrow (a \cup b),\ ((\neg a) \Rightarrow (a \cup b)),\ \neg(a \Rightarrow (a \cup b)), (a \to a)\}$$

1. Determine which of the elements of $\mathcal{S}$ **are**, and which **are not** well formed formulas of $\mathcal{L} = (\mathcal{A}, \mathcal{F})$

2. For any $A \notin \mathcal{F}$ **re-write** it as a correct formula and **write** what it says in the natural language

Exercises

Solution

The formula $\neg a \Rightarrow (a \cup b)$ **is not** a well formed formula

A **correct** formula is $(\neg a \Rightarrow (a \cup b))$

It says: **”If a is not true , then we have a or b ”**

Another **correct** formula in is $\neg(a \Rightarrow (a \cup b))$

It says: **”It is not true that a implies a or b ”**

Exercises

Solution

The formula $((\neg a) \Rightarrow (a \cup b))$ is **not correct** because $(\neg a) \notin \mathcal{F}$

The correct formula is $(\neg a \Rightarrow (a \cup b))$

The formula $\neg(a \Rightarrow (a \cup b))$ is **correct**

The formula $\neg(a \rightarrow a) \notin \mathcal{F}$ is **not correct**

The connective $\rightarrow$ does not belong to the language $\mathcal{L}$

$\neg(a \rightarrow a)$ is a correct formula of **another propositional language**; the one that uses a symbol $\rightarrow$ for implication

Exercise

Write following natural language statement:
"One likes to play bridge or from the fact that the weather is good we conclude the following: one does not like to play bridge or one likes swimming"
as a formula of the propositional language $\mathcal{L} = (\mathcal{A}, \mathcal{F})$

Solution

First we identify the needed components of the alphabet $\mathcal{A}$:

propositional variables: a, b, c

a denotes statement: one likes to play bridge, b denotes a statement: the weather is good, c denotes a statement: one likes swimming

Connectives: $\cup$, $\Rightarrow$, $\cup$. $\neg$

The corresponding **formula** of $\mathcal{L}$ is

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

Symbols for Connectives

The connectives symbols we use are not the only one used in mathematical, logical, or computer science literature

Some Other Symbols

Negation	Disjunction	Conjunction	Implication	Equivalence
$-A$	$(A \cup B)$	$(A \cap B)$	$(A \Rightarrow B)$	$(A \Leftrightarrow B)$
NA	DAB	CAB	IAB	EAB
$\overline{A}$	$(A \vee B)$	$(A \,\&\, B)$	$(A \rightarrow B)$	$(A \leftrightarrow B)$
$\sim A$	$(A \vee B)$	$(A \cdot B)$	$(A \supset B)$	$(A \equiv B)$
A'	$(A + B)$	$(A \cdot B)$	$(A \rightarrow B)$	$(A \equiv B)$

The first notation is the closest to ours and is drawn mainly from the algebra of sets and lattice theory

The second comes from the Polish logician **J. Łukasiewicz** and is called the Polish notation

The third was used by **D. Hilbert.**

The fourth comes from **Peano** and **Russell**

The fifth goes back to **Schröder** and **Pierce**

Chapter 2
Introduction to Classical Logic Languages and Semantics

Slides Set 1

PART 3: Propositional Semantics

Propositional Semantics

We present now **definitions** of propositional connectives in terms of **two logical values** true or false and discuss their **motivations**

The resulting definitions are called a **semantics** for the **classical** propositional connectives

The **semantics** presented here is fairly **informal**

The **formal definition** of **classical** propositional semantics is presented in **chapter 3**

Conjunction: Motivation and Definition

Conjunction

A **conjunction** $(A \cap B)$ is a **true** formula if both A and B are **true** formulas

If one of the formulas, or both, are **false**, then the **conjunction** is a **false** formula

Let's denote statement: "formula A is **false** " by $A = F$ and a statement: "formula A is **true** " by $A = T$

Conjunction: Definition

Conjunction

The logical value of a **conjunction** depends on the logical values of its factors in a way which is express in the form of the following table (truth table)

Conjunction Table

A	B	$(A \cap B)$
T	T	T
T	F	F
F	T	F
F	F	F

Disjunction

The word or is used in natural language in two different senses.

First: A or B is true if at **least one** of the statements A, B is true

Second: A or B is true if **one** of the statements A and B is true and the other is false

In **mathematics** and hence in **logic**, the word or is used in the **first sense**

Disjunction: Definition

Disjunction

We adopt the convention that a **disjunction** $(A \cup B)$ is true if **at least one** of the formulas A , B is true

Disjunction Table

A	B	$(A \cup B)$
T	T	T
T	F	T
F	T	T
F	F	F

Negation: Definition

Negation

The **negation** of a true formula is a false formula, and the negation of a false formula is a true formula

Negation Table

A	$\neg A$
T	F
F	T

Implication: Motivation and Definition

The semantics of the statements in the form

if A, then B

needs a little bit more discussion.

In everyday language a statement *if A, then B* is interpreted to mean that B can be **inferred** from A.

In mathematics its interpretation differs from that in natural language

Implication: Definition

Implication

The above examples **justify** adopting the following definition of a semantics for the implication $(A \Rightarrow B)$

Implication Table

A	B	$(A \Rightarrow B)$
T	T	T
T	F	F
F	T	T
F	F	T

Implication: Motivation

Consider the following

Theorem

For every natural number n,

if 6 DIVIDES n, then 3 DIVIDES n

The theorem is **true** for any natural number, hence in particular, it is **true** for numbers 2, 3, 6

Consider number 2

The following proposition is **true**

if 6 DIVIDES 2, then 3 DIVIDES 2

It means an implication $(A \Rightarrow B)$ in which A and B are **false** is interpreted as a **true** statement

Implication: Motivation

Consider now a number 3

The following proposition **is true**

if 6 DIVIDES 3, then 3 DIVIDES 3,

It means that an implication $(A \Rightarrow B)$ in which A is **false** and B is **true** is interpreted as a **true statement**

Consider now a number 6

The following proposition is **true**

if 6 DIVIDES 6, then 3 DIVIDES 6.

It means that an implication $(A \Rightarrow B)$ in which A and B are **true** is interpreted as a **true statement**

Implication: Motivation

One more case.

What happens when in the implication $(A \Rightarrow B)$ the formula A is **true** and the formula B is **false**

Consider a sentence

if 6 DIVIDES 12, then 6 DIVIDES 5.

Obviously, this is a **false statement**

Equivalence Definition

Equivalence

An equivalence $(A \Leftrightarrow B)$ is **true** if both formulas A and B have the same logical value

Equivalence Table

A	B	$(A \Leftrightarrow B)$
T	T	T
T	F	F
F	T	F
F	F	T

Truth Tables Semantics

We summarize the tables for propositional connectives in the following one table.

We call it a **truth table definition** of propositional; connectives and hence we call the semantics defined here a **truth tables semantics**.

A	B	$\neg A$	$(A \cap B)$	$(A \cup B)$	$(A \Rightarrow B)$	$(A \Leftrightarrow B)$
T	T	F	T	T	T	T
T	F	F	F	T	F	F
F	T	T	F	T	T	F
F	F	T	F	F	T	T

Truth Tables Semantics

The truth tables indicate that the logical value of of propositional connectives **independent** of the formulas A, B

We write the connectives in a **"formula independent"** form as a set of of the following **equations**

$$\neg T = F, \quad \neg F = T;$$

$$T \cap T = T, \quad T \cap F = F, \quad F \cap T = F, \quad F \cap F = F;$$

$$T \cup T = T, \quad T \cup F = T, \quad F \cup T = T, \quad F \cup F = F;$$

$$T \Rightarrow T = T, \quad T \Rightarrow F = F, \quad F \Rightarrow T = T, \quad F \Rightarrow F = T;$$

$$T \Leftrightarrow T = T, \quad T \Leftrightarrow F = F, \quad F \Leftrightarrow T = F, \quad T \Leftrightarrow T = T$$

We use the above set of **connectives equations** to evaluate **logical values** of formulas

Exercise

Exercise

Show that $(A \Rightarrow (\neg A \cap B)) = F$ for the following logical values of its basic components: A=T and B=F

Solution

We **calculate** the logical value of the formula

$$(A \Rightarrow (\neg A \cap B))$$

by **substituting** the respective logical values T, F for the component formulas A, B and applying the set of **connectives equations** as follows

$$T \Rightarrow (\neg T \cap F) = T \Rightarrow (F \cap F) = T \Rightarrow F = F$$

Extensional Connectives

Extensional connectives are the connectives that have the following property:

the logical value of the formulas form by means of these connectives and certain given formulas depends only on the logical value(s) of the given formulas

All classical **propositional connectives**

$$\neg, \ \cup, \ \cap, \ \Rightarrow, \ \Leftrightarrow$$

are **extensional**

Propositional Connectives

Remark

In everyday language there are expressions such as

"I believe that", "it is possible that", " certainly", etc....

They are represented by some propositional connectives which are not extensional

They do not play any role in **mathematics** and so are not discussed in **classical logic**, they belong to **non-classical logics**

All Extensional Two Valued Connectives

There are many **other binary** (two valued) **extensional** propositional connectives

Here is a table of **all unary** connectives

A	$\triangledown_1 A$	$\triangledown_2 A$	$\neg A$	$\triangledown_4 A$
T	F	T	F	T
F	F	F	T	T

All Extensional Binary Connectives

Here is a table of **all binary connectives**

A	B	$(A \circ_1 B)$	$(A \cap B)$	$(A \circ_3 B)$	$(A \circ_4 B)$
T	T	F	T	F	F
T	F	F	F	T	F
F	T	F	F	F	T
F	F	F	F	F	F
A	B	$(A \downarrow B)$	$(A \circ_6 B)$	$(A \circ_7 B)$	$(A \Leftrightarrow B)$
T	T	F	T	T	T
T	F	F	T	F	F
F	T	F	F	T	F
F	F	T	F	F	T
A	B	$(A \circ_9 B)$	$(A \circ_{10} B)$	$(A \circ_{11} B)$	$(A \cup B)$
T	T	F	F	F	T
T	F	T	T	F	T
F	T	T	F	T	T
F	F	F	T	T	F
A	B	$(A \circ_{13} B)$	$(A \Rightarrow B)$	$(A \uparrow B)$	$(A \circ_{16} B)$
T	T	T	T	F	T
T	F	T	F	T	T
F	T	F	T	T	T
F	F	T	T	T	T

Functional Dependency Definition

Definition

Functional dependency of connectives is the ability of defining **some** connectives in terms of some **others**

All classical propositional connectives can be **defined** in terms of disjunction and negation

Two binary connectives: $\downarrow$ and $\uparrow$ suffice, each of them separately, to **define** all classical connectives, whether unary or binary

Functional Dependency

The connective $\uparrow$ was discovered in 1913 by **H.M. Sheffer**, who called it **alternative negation**

Now it is often called a Sheffer's connective

The formula

$A \uparrow B$ **reads**: not both A and B.

Negation $\neg A$ is defined as $A \uparrow A$.

Disjunction $(A \cup B)$ is defined as $(A \uparrow A) \uparrow (B \uparrow B)$

Functional Dependency

The connective $\downarrow$ was discovered by **J. Łukasiewicz** and is called a **joint negation**

The formula

$A \downarrow B$ **reads**: neither A nor B.

It was proved in 1925 by **E. Żyliński** that **no** propositional connective other than $\uparrow$ and $\downarrow$ **suffices** to define all the remaining classical connectives

Chapter 2
Introduction to Classical Logic Languages and Semantics

Slides Set 1

PART 4: Examples of Propositional Tautologies

Propositional Tautologies

Now we connect syntax (formulas of a given language $\mathcal{L}$) with semantics (assignment of truth values to the formulas of the language $\mathcal{L}$)

In **logic** we are interested in those propositional formulas that must be] **always true** because of their syntactical structure without reference to the natural language meaning of the propositions they **represent**

Such formulas are called **propositional tautologies**

Example

Example

Given a formula $(A \Rightarrow A)$

We evaluate the logical value of our formula for **all possible** logical values of its basic component A

We put our **calculation** in a form of a table, called a **truth table** below

A	$(A \Rightarrow A)$ computation	$(A \Rightarrow A)$
T	$T \Rightarrow T = T$	**T**
F	$F \Rightarrow F = T$	**T**

The **logical value** of the formula $(A \Rightarrow A)$ is **always** T

This means that it is a **propositional tautology**.

Example

Example

Here is a **truth table** for a formula $(A \Rightarrow B)$

A	B	$(A \Rightarrow B)$ computation	$(A \Rightarrow B)$
T	T	$T \Rightarrow T = T$	**T**
T	F	$T \Rightarrow F = F$	**F**
F	T	$F \Rightarrow T = T$	**T**
F	F	$F \Rightarrow F = T$	**T**

The **logical value** of the formula $(A \Rightarrow B)$ is F for A = T and B = F what means that **it is not** a **propositional tautology**

Definition

For any formula $A \in \mathcal{F}$ of a propositional language $\mathcal{L} = (\mathcal{A}, \mathcal{F})$, we say that A is a **propositional tautology**

if and only if

the logical value of A is T (we write it A = T) for all possible logical values of its **basic** components

We write

$$\models A$$

to denote that A is a **tautology**

Classical Tautologies

Here is a **list** of some of the most known classical **notions** and **tautologies**

Modus Ponens known to the Stoics (3rd century B.C.)

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

Detachment

$$\models ((A \cap (A \Leftrightarrow B)) \Rightarrow B)$$

$$\models ((B \cap (A \Leftrightarrow B)) \Rightarrow A)$$

Sufficient and Necessary

Sufficient: Given an implication $(A \Rightarrow B)$,
A is called a **sufficient** condition for B to hold.

Necessary : Given an implication $(A \Rightarrow B)$,
B is called a **necessary** condition for A to hold.

Implication Names

Simple:

$(A \Rightarrow B)$ is called a simple implication

Converse:

$(B \Rightarrow A)$ is called a converse implication to $(A \Rightarrow B)$

Opposite:

$(\neg B \Rightarrow \neg A)$ is called an opposite implication to $(A \Rightarrow B)$

Contrary:

$(\neg A \Rightarrow \neg B)$ is called a contrary implication to $(A \Rightarrow B)$

Laws of Contraposition

$$\models ((A \Rightarrow B) \Leftrightarrow (\neg B \Rightarrow \neg A)),$$

$$\models ((B \Rightarrow A) \Leftrightarrow (\neg A \Rightarrow \neg B)).$$

These Laws make it possible to **replace**, in any deductive argument, a sentence of the form $(A \Rightarrow B)$ by $(\neg B \Rightarrow \neg A)$, and **conversely**

Necessary and sufficient

We read the formula $(A \Leftrightarrow B)$ as

"B **is necessary and sufficient for** A"

because of the following tautology

$$\models ((A \Leftrightarrow B)) \Leftrightarrow ((A \Rightarrow B) \cap (B \Rightarrow A)))$$

Stoics, 3rd century B.C.

Hypothetical Syllogism

$$\models (((A \Rightarrow B) \cap (B \Rightarrow C)) \Rightarrow (A \Rightarrow C)),$$

$$\models ((A \Rightarrow B) \Rightarrow ((B \Rightarrow C) \Rightarrow (A \Rightarrow C))),$$

$$\models ((B \Rightarrow C) \Rightarrow ((A \Rightarrow B) \Rightarrow (A \Rightarrow C))).$$

Modus Tollendo Ponens

$$\models (((A \cup B) \cap \neg A) \Rightarrow B),$$

$$\models (((A \cup B) \cap \neg B) \Rightarrow A)$$

12 to 19 Century

Duns Scotus 12/13 century

$$\models (\neg A \Rightarrow (A \Rightarrow B))$$

Clavius 16th century

$$\models ((\neg A \Rightarrow A) \Rightarrow A)$$

Frege 1879

$$\models (((A \Rightarrow (B \Rightarrow C)) \cap (A \Rightarrow B)) \Rightarrow (A \Rightarrow C)),$$

$$\models ((A \Rightarrow (B \Rightarrow C)) \Rightarrow ((A \Rightarrow B) \Rightarrow (A \Rightarrow C)))$$

Frege gave the the **first** formulation of the classical propositional logic as a formalized axiomatic system

Apagogic Proofs

Apagogic Proofs: means proofs by reductio ad absurdum

Reductio ad absurdum:

to prove A to be **true**, we assume $\neg A$. If we get a contradiction, it means that we have proved A to be **true**

Correctness of this reasoning is guarantee by the following **tautology**

$$\models ((\neg A \Rightarrow (B \cap \neg B)) \Rightarrow A)$$

Classical Tautologies

This chapter contains a very extensive list of **classical propositional tautologies**

Read, prove , and **memorize** as many as you can

We will use them freely in later Chapters assuming that you are really familiar with all of them

Chapter 2
Introduction to Classical Logic Languages and Semantics

Slides Set 2

PART 5: Predicate Language

Predicate Language

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

The **predicate language** is much more complicated in its structure then the propositional one

Its alphabet $\mathcal{A}$ is much richer.
The definition of its set of formulas $\mathcal{F}$ is more complicated

Predicate Language

In order to **define** the set $\mathcal{F}$ define an additional set **T**, called a set of all **terms** of the predicate language $\mathcal{L}$

We **single** out this set **T** of **terms** not only because we need it for the definition of formulas, but also because of its role in the **development** of other notions of **predicate logic**.

Predicate Language Definition

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, and $\mathcal{F}$ is a set of formulas

Alphabet Components

Alphabet $\mathcal{A}$

The components of $\mathcal{A}$ are as follows

1. Propositional connectives

$$\neg, \ \cap, \ \cup, \ \Rightarrow, \ \Leftrightarrow$$

2. Quantifiers $\forall, \ \exists$

$\forall$ is the universal quantifier, and $\exists$ is the existential quantifier

3. Parenthesis (and)

Alphabet Components

4. Variables

We assume that we have, as we did in the propositional case a countably infinite set VAR of variables

The variables now have a **different meaning** than they had in the propositional case

We hence call them variables, or individual variables

We put

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

5. Constants

The constants represent in ”real life” elements of concrete sets We assume that we have a countably infinite set C of constants

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

Alphabet Components

6. Predicate symbols

The predicate symbols represent ”real life” relations

We denote them by P, Q, R, ..., with indices, if necessary

We use symbol **P** for the set of all predicate symbols

We assume that **P** is countably infinite and write

$$\mathbf{P} = \{P_1, P_2, P_3, ...\}$$

Alphabet Components

Logic notation

In "real life" we write symbolically $x < y$ to express that element x is smaller then element y according to the two argument order relation $<$

In the **predicate language** $\mathcal{L}$ we represent the relation $<$ as a two argument predicate $P \in \mathbf{P}$

We write $P(x, y)$ as a **representation** of "real life" $x < y$.

The variables x, y in $P(x, y)$ are **individual variables** from the set VAR

Mathematical statements $n < 0$, $1 < 2$, $0 < m$ are **represented** in $\mathcal{L}$ by $P(x, c_1)$, $P(c_2, c_3)$, $P(c_1, y)$, respectively,

where c_1, c_2, c_3 are any **constants** and x, y any **variables**

Alphabet Components

7. Function symbols

The function symbols represent ”real life” functions

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

We use symbol **F** for the set of all function symbols

We assume that **F** is countably infinite and write

$$\mathbf{F} = \{f_1, f_2, f_3, ...\}$$

Set **T** of Terms

Definition

Terms are expressions built out of function symbols and variables

Terms describe how we build compositions of functions

We define the set **T** of all **terms** recursively as follows.

1. All variables are **terms**;

2. All constants are **terms**;

3. For any **function symbol** $f \in$ **F** representing a function on n variables, and any **terms** $t_1, t_2, ..., t_n$, the expression $f(t_1, t_2, ..., t_n)$ is a **term**;

4. The set **T** of all **terms** of the predicate language $\mathcal{L}$ is the smallest set that fulfills the conditions **1. - 3.**

Example

Example

Here are some terms of $\mathcal{L}$

$$h(c_1),\ f(g(c,x)),\ g(f(f(c)),g(x,y)),$$

$$f_1(c,g(x,f(c))),\ g(g(x,y),g(x,h(c)))\ \ldots.$$

Observe that to obtain the predicate language **representation** of for example $x+y$ we can first write it as $+(x,y)$ and then replace the addition symbol $+$ by any two argument function symbol $g \in \mathbf{F}$ and get the **term** $g(x,y)$.

Set $\mathcal{F}$ of Formulas

Formulas are build out of elements of the **alphabet** $\mathcal{A}$ and the set **T** of all **terms**

We denote the **formulas** by $A, B, C, ...$, with indices, if necessary.

We them, as before in **recursive steps**

The **first** recursive step says:

all atomic formulas are **formulas**

The atomic formulas are the simplest formulas, as the propositional variables were in the case of the **propositional** language.

We define the atomic formulas as follows.

Atomic Formulas

Definition

An **atomic formula** is any expression of the form

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

where R is any n-argument predicate $R \in \mathbf{P}$ and $t_1, t_2, ..., t_n$ are terms, i.e. $t_1, t_2, ..., t_n \in \mathbf{T}$.

Some **atomic formulas** of $\mathcal{L}$ are:

$$Q(c),\ Q(x),\ Q(g(x_1, x_2)),$$

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

Set $\mathcal{F}$ of Formulas

Definition

The set $\mathcal{F}$ of formulas of predicate language $\mathcal{L}$ is the smallest set meeting the following conditions

1. All **atomic** formulas are **formulas**;

2. If A, B are **formulas**, then $\neg A, (A \cap B), (A \cup B), (A \Rightarrow B), (A \Leftrightarrow B)$ are **formulas**;

3. If A is a **formula**, then $\forall x A$, $\exists x A$ are **formulas** for any variable $x \in VAR$.

Set $\mathcal{F}$ of Formulas

Example

Some formulas of $\mathcal{L}$ are:

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

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

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

Set $\mathcal{F}$ of Formulas

Let's look now closer at the following formulas.

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

$$\exists x R(x, y), \quad \forall y R(x, y), \quad \exists x \forall y R(x, y).$$

Observations

1. Some formulas are **without quantifiers**:

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

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

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

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

Set $\mathcal{F}$ of Formulas

Observations

2. Quantifiers bind variables within formulas.

The variable x is **bounded** by $\exists x$ in the formula

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

the variable y is **free**

The variable y is **bounded** by $\forall y$ in the formula

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

the variable y is **free**.

Set $\mathcal{F}$ of Formulas

Observations

3. The formula

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

does not contain any **free** variables, **neither does** the formula

$$R(c_1, c_2)$$

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

Mathematical Statements

We often use **logic symbols**, while writing mathematical statements in a symbolic way

For **example**, mathematicians to say

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

and often write it as

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

Mathematical Statements

Some mathematicians who are more "logic oriented" would write the satements as follows

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

or even write it as

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

Observe that none of the above symbolic statement are correct formulas of the **predicate language**.

These are mathematical statements 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**

Mathematical Statements

Our **goal** now is to "translate " mathematical and natural language statement into correct **formulas** of the predicate language $\mathcal{L}$

Let's start with some **observations**

O1 The quantifiers in $\forall_{x \in N}, \ \exists_{y \in Z}$ often used by mathematicians **are not** the one defined and used in **logic**

O2 The predicate language $\mathcal{L}$ admits **only** quantifiers $\forall x, \ \exists y,$ for any variables $x, y \in VAR$

Quantifiers with Restricted Domain

O3 The quantifiers $\forall_{x \in N}$, $\exists_{y \in Z}$ are called quantifiers with **restricted** domain, or **restricted** domain quantifiers

Definition

$\forall_{A(x)} B(x)$ stands for a formula $\forall x(A(x) \Rightarrow B(x)) \in \mathcal{F}$

$\exists_{A(x)} B(x)$ stands for a formula $\exists x(A(x) \cap B(x)) \in \mathcal{F}$

The **restriction** of the quantifier domain can, and often is given by more complicated statements

Quantifiers with Restricted Domain

We write the definition of the restricted domain quantifiers in a form of the following rules

Transformations Rules for **Restricted Quantifiers**

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

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

Translations to Formulas of $\mathcal{L}$

Given a mathematical statement **S** written with the use of logical symbols.

We obtain a formula $A \in \mathcal{F}$ that is a **translation** of **S** into the predicate language $\mathcal{L}$ by conducting a following sequence of 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**

Step 2 We **write** the basic statements as **atomic formulas** of the predicate language $\mathcal{L}$

Translations to Formulas of $\mathcal{L}$

Step 3 We **write** the statement **S** a formula with restricted quantifiers (if needed)

Step 4 We **apply** the transformations rules for restricted quantifiers to the formula obtained in the **Step 3**

In case of a translation from mathematical statement **S** written **without** logical symbols **we add** a following step

Step 0 We **identify** propositional connectives and quantifiers and use them to re-write the statement in a form that is as close to the structure of a logical formula as possible

Translations to Formulas of $\mathcal{L}$

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 proceed as follows

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

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

Translations to Formulas of $\mathcal{L}$

Step 2 We **write** the basic statements as **atomic formulas** of the predicate language $\mathcal{L}$

Remember that in the predicate language $\mathcal{L}$ we write a function symbol in front of the function arguments not between them as we write in mathematics

The same applies to relation symbols

Translations to Formulas of $\mathcal{L}$

Example

We re-write a basic mathematical statement

$$x + 2 > y \quad \text{as} \quad > (+(x, 2), y),$$

and then we write it as an **atomic formula**

$$P(f(x, c), y)$$

$P \in \mathbf{P}$ stands for two argument relation $>$

$f \in \mathbf{F}$ stands for two argument function $+$

$c \in \mathbf{C}$ stands for the number 2

Translations to Formulas of $\mathcal{L}$

Step 3 We **write** the statement **S** a formula with restricted quantifiers (if needed)

Step 4 We **apply** the transformations rules for restricted quantifiers to the formula obtained in the **Step 3**

In case of a translation from mathematical statement written without logical symbols **we add** a following step.

Step 0 We **identify** propositional connectives and quantifiers and use them to re-write the statement in a form that is as close to the structure of a logical formula as possible

Translations Examples

Exercise

Given a mathematical statement **S** written with logical symbols

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

1. Translate it into a proper **logical** formula with restricted quantifiers i.e. into a formula of $\mathcal{L}$ that **uses** the restricted domain quantifiers.

2. Translate your restricted quantifiers formula into a correct formula **without** restricted domain quantifiers, i.e. into a proper formula of $\mathcal{L}$

A **long** and **detailed** solution is given in Chapter 2, page 28.

A **short** statement of the exercise and a **short** solution follows

Translations Examples

Exercise

Given a mathematical statement **S** written with logical symbols

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

Translate it into a proper formula of $\mathcal{L}$

Short Solution

The basic statements in **S** are:

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

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

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

Translations Examples

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

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

By the **Transformation Rules** we get the formula $A \in \mathcal{F}$

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

Translations Examples

Exercise

Here is a mathematical statement **S**

”For all real numbers x the following holds: If $x < 0$, then there is a natural number n, such that $x + n < 0$.”

1. Re-write **S** as a symbolic mathematical statement SF that only uses mathematical and logical symbols.

2. Translate the symbolic statement SF into to a corresponding formula $A \in \mathcal{F}$ of the predicate language $\mathcal{L}$

Translations Examples

Solution

The statement **S** is

"For all real numbers x the following holds: If $x < 0$, then there is a natural number n, such that $x + n < 0$."

S becomes a symbolic mathematical statement SF

$$\forall_{x \in R}(x < 0 \Rightarrow \exists_{n \in N}\, x + n < 0)$$

We write R(x) for $x \in R$, N(y) for $n \in N$, a constant c for the number 0. We use $L \in \mathbf{P}$ to denote the relation $<$. We use $f \in \mathbf{F}$ to denote the function $+$

The statement $x < 0$ becomes an **atomic** formula

$$L(x, c)$$

Translations Examples

The statement $x + n < 0$ becomes an **atomic formula**

$$L(f(x, y), c)$$

The symbolic mathematical statement SF

$$\forall_{x \in R}(x < 0 \Rightarrow \exists_{n \in N}\, x + n < 0)$$

becomes a **restricted quantifiers** formula

$$\forall_{R(x)}(L(x, c) \Rightarrow \exists_{N(y)} L(f(x, y), c))$$

We apply now the **transformation rules** and get a corresponding formula $A \in \mathcal{F}$

$$\forall x(N(x) \Rightarrow (L(x, c) \Rightarrow \exists y(N(y) \cap L(f(x, y), c)))$$

Translations from Natural Language

Exercise

Translate a natural language statement **S**

"Any friend of Mary is a friend of John and Peter is not John's friend. Hence Peter is not May's friend"

into a formula $A \in \mathcal{F}$ of the predicate language $\mathcal{L}$

Solution

Step 1 We identify the basic relations and functions (if any) and **translate** them into **atomic** formulas

We have only one relation of "being a friend"

We **translate** it into an **atomic** formula

$$F(x, y),$$

where F(x, y) stands for "x is a friend of y"

Translations from Natural Language

"Any friend of Mary is a friend of John and Peter is not John's friend. Hence Peter is not May's friend"

We use **constants** m, j, p for Mary, John, and Peter, respectively

Step 2 We hence have the following **atomic formulas**:

$$F(x, m), \quad F(x, j), \quad F(p, j)$$

where F(x, m) stands for "x is a friend of Mary",

F(x, j) stands for "x is a friend of John", and

F(p, j) stands for "Peter is a friend of John"

Translations from Natural Language

Step 3 Statement "Any friend of Mary is a friend of John" **translates** into a restricted quantifier formula

$$\forall_{F(x,m)} F(x,j)$$

Statement "Peter is not John's friend" **translates** into

$$\neg F(p,j)$$

and "Peter is not May's friend" **translates** into

$$\neg F(p,m)$$

Translations from Natural Language

Restricted quantifiers formula for **S** is

$$((\forall_{F(x,m)} F(x,j) \cap \neg F(p,j)) \Rightarrow \neg F(p,m))$$

4 By the **Transformation Rules**, the formula $A \in \mathcal{F}$ of $\mathcal{L}$ corresponding to **S** is

$$((\forall x(F(x,m) \Rightarrow F(x,j)) \cap \neg F(p,j)) \Rightarrow \neg F(p,m))$$

Rules of Translations

Rules of translation from natural language to the predicate language $\mathcal{L}$

Given a statement **S**

1. Identify the basic relations and functions (if any) and **translate** them into **atomic** formulas

2. Identify propositional connectives and use symbols $\neg, \cup, \cap, \Rightarrow, \Leftrightarrow$ for them

3. Identify quantifiers: restricted $\forall_{A(x)},\ \exists_{A(x)}$, and non-restricted $\forall x,\ \exists x$

4. Use the symbols from **1.** - **3.** and write logic formula containing restricted and non-restricted quantifiers, if any

5. Use the restricted quantifiers **Transformation Rules** to write $A \in \mathcal{F}$ of the predicate language $\mathcal{L}$ corresponding to **S**

Translation Example

Exercise

Given a natural language statement

S: "For any bird one can find some birds that white"

Show that the **translation** of **S** into a formula of the predicate language $\mathcal{L}$ is

$$\forall x(B(x) \Rightarrow \exists x(B(x) \cap W(x)))$$

Solution

We follow the **Rules of Translation** as flollows

1. Atomic formulas: B(x), W(x)

where B(x) stands for " x is a bird"

and W(x) stands for " x is white"

Translation Example

2. There is no propositional connectives in **S**

3. Restricted quantifiers:

$\forall_{B(x)}$ for "any bird "

$\exists_{B(x)}$ for "one can find some birds"

4. Restricted quantifiers formula for **S** is

$$\forall_{B(x)}\exists_{B(x)}\ W(x)$$

5. By the **Transformation Rules** we get a required formula of the predicate language $\mathcal{L}$:

$$\forall x(B(x) \Rightarrow \exists x(B(x) \cap W(x)))$$

Translation Example

Exercise

Translate into $\mathcal{L}$ a natural language statement
S: " Some patients like all doctors"

Solution

1. Atomic formulas: P(x), D(x), L(x, y)

P(x) stands for " x is a patient",

D(x) stands for " x is a doctor", and

L(x,y) stands for " x likes y"

2. There is no propositional connectives in **S**

Translation Example

3. Restricted quantifiers:

$\exists_{P(x)}$ for "some patients " and $\forall_{D(x)}$ for "all doctors"

Observe that we **can't** write L(x, D(y)) for "x likes doctor y"

D(y) is a **predicate**, not a **term**, and hence L(x, D(y)) is not a **formula**

We have to **express** the statement " x likes all doctors y" in terms of restricted quantifiers and the predicate L(x,y) only

Translation Example

Observe that the statement " x likes all doctors y" means also " all doctors y are liked by x"

We hence **re- write** it as "for all doctors y, x likes y" what translates to a formula

$$\forall_{D(y)} L(x, y)$$

Hence the statement **S** translates to

$$\exists_{P(x)} \forall_{D(x)} L(x, y)$$

4. By the **Transformation Rules** we get the following **translation** of **S** into $\mathcal{L}$

$$\exists x (P(x) \cap \forall y (D(y) \Rightarrow L(x, y)))$$

Chapter 2
Introduction to Classical Logic Languages and Semantics

Slides Set 3

PART 6: Predicate Tautologies - Laws for Quantifiers

Predicate Tautologies

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

We **introduce** it intuitively here and **define** it formally in chapter 8

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

We provide here a motivation, some examples and intuitive definitions

We also list and discuss the most used and useful **predicate tautologies** and **equational laws** of quantifiers

Interpretation

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

We **define** the **interpretation** I in a set $U \neq \emptyset$ by interpreting predicate and functional **symbols** of $\mathcal{L}$ as concrete **relations** and **functions** defined in the set U

We interpret constants symbols as **elements** of the set U

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

Model Structure

We define a **model structure** for the predicate language $\mathcal{L}$ as a pair

$$\mathbf{M} = (U, \; I)$$

where the set U is called the structure **universe** and of the **I** is the structure **interpretation** in the universe U

Given a formula A of $\mathcal{L}$, and the **model structure** $\mathbf{M} = (U, I)$

We **denote** by

$$A_I$$

a statement defined in the structure $\mathbf{M} = (U, I)$ that is **determined** by the formula A and the interpretation I in the universe U

Model Structure

When the formula A is a **sentence**, it means it is a formula without free variables, the **model structure** statement

$$A_I$$

represents a proposition that is **true** or **false** in the universe U, under the interpretation I

When the formula A **is not** a sentence, it contains free variables and may be **satisfied** (i.e. true) for some values in the universe U and **not satisfied** (i.e. false) for the others

Lets look at few simple examples

Examples

Example

Let A be a formula $\exists x P(x, c)$

Consider a **model structure** $\mathbf{M}_1 = (N, I_1)$

The universe of the interpretation I_1 is the set N of natural numbers

We **define** I_1 as follows:

We **interpret** the two argument predicate P as a relation $<$ and the constant c as number 5, i.e we put

$P_{I_1} : =$ and $c_{I_1} : 5$

Examples

The formula A: $\exists x P(x, c)$ under the interpretation I_1 becomes a mathematical statement

$$\exists x \ x < 0$$

defined in the set N of natural numbers

We write it for short

$$A_{I_1} : \ \exists_{x \in N} \ x = 5$$

A_{I_1} is obviously a **true** mathematical statement in the model structure $\mathbf{M}_1 = (N, I_1)$

We write it **symbolically** as

$$\mathbf{M}_1 \models \exists x P(x, c)$$

and say: $\mathbf{M}_1$ is a **model** for the formula A

Examples

Example

Consider now a model structure $\mathbf{M}_2 = (N, I_2)$ and the formula A: $\exists x P(x, c)$

We **interpret** now the predicate P as relation $<$ in the set N of natural numbers and the constant c as number 0

We write it as

$$P_{I_2} : < \quad \text{and} \quad c_{I_2} : \ 0$$

Examples

The formula A: $\exists x P(x, c)$ under the interpretation I_2 becomes a mathematical statement $\exists x \; x < 0$ defined in the set N of natural numbers

We write it for short

$$A_{I_2} : \quad \exists_{x \in N} \; x < 0$$

A_{I_2} is obviously a **false** mathematical statement.

We say: the formula A: $\exists x P(x, c)$ is **false** under the interpretation I_2 in $\mathbf{M}_2$, or we say for short: A is **false** in $\mathbf{M}_2$

We write it **symbolically** as

$$\mathbf{M}_2 \not\models \; \exists x P(x, c)$$

and say that $\mathbf{M}_2$ is a **counter-model** for the formula A

Example

Consider now a **model structure**

$\mathbf{M}_3 = (Z, I_3)$ and the formula A: $\exists x P(x, c)$

We **define** an interpretation I_3 in the set of all integers Z exactly as the interpretation I_1 was defined, i.e. we put

$$P_{I_3} : < \quad \text{and} \quad c_{I_3} : 0$$

Examples

In this case we get

$$A_{I_3} : \ \exists_{x \in Z}\ x < 0$$

Obviously A_{I_3} is a **true** mathematical statement

The formula A is **true** under the interpretation I_3 in $\mathbf{M}_3$ (A is **satisfied, true** in $\mathbf{M}_3$)

We write it symbolically as

$$\mathbf{M}_3 \models \ \exists x P(x, c)$$

$\mathbf{M}_3$ is yet another **model** for the formula A

Examples

When a formula A **is not** a closed, i.e. is not a sentence, the situation gets more complicated

A can be **satisfied** (i.e. true) for some values in the universe U of a $\mathbf{M} = (U, I)$

But also and can be **not satisfied** (i.e. false) for some other values in the universe U of a $\mathbf{M} = (U, I)$

We explain it in the following examples

Examples

Example

Consider a formula

$$A_1 : R(x, y),$$

We define a model structure

$$\mathbf{M} = (N, I)$$

where R is **interpreted** as a relation $\leq$ defined in the set N of all natural numbers, i.e. we put $R_I : \leq$

In this case we get

$$A_{1I} : \ x \leq y$$

and $A_1 : \ R(x, y)$ is **satisfied** in model structure $\mathbf{M} = (N, I)$ by all $n, m \in N$ such that $n \leq m$

Example

Consider a following formula

$$A_2 : \ \forall y R(x, y)$$

and the same model structure $\mathbf{M} = (N, I)$, where R is **interpreted** as a relation $\leq$ defined in the set N of all natural numbers, i.e. we put

$$R_I : \leq$$

In this case we get that

$$A_{2I} : \forall_{y \in N} \ x \leq y$$

and so the formula $A_2 : \ \forall y R(x, y)$ is **satisfied** in $\mathbf{M} = (N, I)$ **only** by the natural number 0

Examples

Example

Consider now a formula

$$A_3 : \exists x \forall y R(x, y)$$

and the same model structure $\mathbf{M} = (N, I)$, where R is **interpreted** as a relation $\leq$ defined in the set N of all natural numbers, i.e. we put $R_I : \leq$

In this case the statement

$$A_{3I} : \exists_{x \in N} \forall_{y \in N}\ x \leq y$$

asserts that there is a smallest number

This is a **true** statement and we call the structure $\mathbf{M} = (N, I)$ ia **model** for the formula $A_3 : \exists x \forall y R(x, y)$

Predicate Tautology Definition

We want the predicate language **tautologies** to have the same property as the **tautologies** of the propositional language, namely to be **always true**

In this case, we intuitively agree that it means that we want the **predicate tautologies** to be formulas that are **true** under **any** interpretation in **any** possible universe

A rigorous definition of the **predicate tautology** is provided in Chapter 8

Predicate Tautology Definition

We construct the rigorous definition of a **predicate tautology** in a following sequence of steps

S1 We define **formally** the notion of **interpretation** I of symbols of the language $\mathcal{L}$ in a set $U \neq \emptyset$, i.e. in a **model structure** $\mathbf{M} = (U, I)$ for $\mathcal{L}$

S2 We define **formally** a notion

" a formula A of $\mathcal{L}$ is **true** in the structure $\mathbf{M} = (U, I)$"

We write it symbolically $\mathbf{M} \models A$ and call thestructure $\mathbf{M} = (U, I)$ a **model** for the formula A

Predicate Tautology Definition

S3 We define a notion "A is a predicate tautology" as follows

Defintion

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

A is a **predicate tautology** (valid formula) if and only if

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

for all model structures $\mathbf{M} = (U, I)$ for the language $\mathcal{L}$

Predicate Tautology Definition

Directly from the above definition we get the following definition of a notion " A is not a predicate tautology"

Defintion

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

A **is not** a predicate **tautology** if and only if

there is a model structure $\mathbf{M} = (U, I)$ for $\mathcal{L}$, such that

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

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

Predicate Tautology Definition

The definition of a notion

" A is not a predicate tautology"

says that in order to prove that a formula A **is not** a predicate tautology one has to show a **counter- model** for it

It means that one has to **define** a non-empty set U and **define** an interpretation I, such that we can prove that

$$A_I$$

is **false**

Predicate Tautology Definition

We use terms **predicate** tautology or **valid** formula instead of just saying a **tautology** in order to distinguish tautologies belonging to two very different languages

For the same reason we usually reserve the symbol $\models$ for **propositional** case

Sometimes we use symbols

$$\models_p \quad \text{or} \quad \models_f$$

to denote **predicate** tautologies

p stands for **predicate** and f stands **first order**

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

We will use **both** names

Predicate Tautologies Examples

Here are some examples of **predicate** tautologies and **counter models** for formulas that are **not** tautologies

Example

For any formula $A(x)$ with a free variable x:

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

Observe that the formula

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

represents an infinite number of formulas.

It is a **tautology** for **any** formula $A(x)$ of $\mathcal{L}$ with a free variable x

Predicate Tautologie Examples

The **inverse** implication to $(\forall x\, A(x) \Rightarrow \exists x\, A(x))$ **is not** a predicate tautology, i.e.

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

To **prove it** we have to provide an **example** of a **concrete formula** $A(x)$ and construct a **counter-model** $\mathbf{M} = (U, I)$ for the formula

$$F : (\exists x\, A(x) \Rightarrow \forall x\, A(x))$$

Let the **concrete** $A(x)$ be an **atomic** formula $P(x, c)$

We define $\mathbf{M} = (N, I)$ for N set of natural numbers and
$P_I : <, \quad c_I : \ 3$

The formula F becomes an obviously **false** mathematical statement

$$F_I : (\exists_{n \in N} n < 3 \Rightarrow \forall_{n \in N} n < 3)$$

Restricted Quantifiers Laws

We have to be very careful when we deal with **restricted domain** quantifiers

For example, the most basic predicate tautology

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

fails when written with the **restricted domain** quantifiers, i.e.

We show that

$$\not\models_p \ (\forall_{B(x)}\, A(x) \Rightarrow \exists_{B(x)}\, A(x))$$

To **prove** this we have to show that corresponding formula of $\mathcal{L}$ obtained by the restricted quantifiers transformations rules **is not** a predicate tautology, i.e. to prove:

$$\not\models_p \ (\forall x(B(x) \Rightarrow A(x)) \Rightarrow \exists x(B(x) \cap A(x))).$$

Restricted Quantifiers Laws

We construct a **counter-model** $\mathbf{M}$ for the formula

$$F: \quad (\forall x(B(x) \Rightarrow A(x)) \Rightarrow \exists x(B(x) \cap A(x)))$$

We take

$$\mathbf{M} = (N, I),$$

where N is the set of natural numbers

We take as the **concrete** formulas $B(x)$, $A(x)$ atomic formulas

$$Q(x,\ c) \quad \text{and} \quad P(x,\ c),$$

respectively, and the interpretation I i defined as

$$Q_I : <, \qquad P_I : >, \quad c_I :$$

Restricted Quantifiers Laws

The formula

$$F: \quad (\forall x(B(x) \Rightarrow A(x)) \Rightarrow \exists x(B(x) \cap A(x)))$$

becomes a **mathematical statement**

$$F_I: \quad (\forall_{n \in N} (x < 0 \Rightarrow n > 0) \Rightarrow \exists_{n \in N}(n < 0 \cap n > 0))$$

The satement F_I is a **false**

because the statement $n < 0$ is **false** for all natural numbers and the implication *false* $\Rightarrow B$ is **true** for any logical value of B

Hence $\forall_{n \in N} (n < 0 \Rightarrow n > 0)$ is a **true** statement and $\exists_{n \in N}(n < 0 \cap n > 0)$ is obviously **false**

Restricted Quantifiers Laws

Restricted quantifiers law corresponding to the predicate tautology

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

is

$$\models_p \; (\forall_{B(x)}\, A(x) \Rightarrow (\exists x\, B(x) \Rightarrow \exists_{B(x)}\, A(x)))$$

We remind that it means that we prove that the corresponding proper formula of $\mathcal{L}$ obtained by the restricted quantifiers **transformations rules** is a predicate tautology, i.e. that

$$\models_p (\forall x(B(x) \Rightarrow A(x)) \Rightarrow (\exists x\, B(x) \Rightarrow \exists x\, (B(x) \cap A(x))))$$

Quantifiers Laws

Another basic **predicate tautology** called a dictum de omni law is

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

where $A(x)$ are any formulas with a free variable x and $y \in VAR$

The corresponding **restricted quantifiers law** is:

$$\models_p (\forall_{B(x)}\, A(x) \Rightarrow (B(y) \Rightarrow A(y))),$$

where $A(x),\ B(x)$ are any formulas with a free variable x and $y \in VAR$

Quantifiers Laws

The next important laws are the **Distributivity Laws**

Distributivity of existential quantifier over conjunction holds only in **one direction**, namely the following is a predicate tautology

$$\models_p \ (\exists x\ (A(x) \cap B(x)) \ \Rightarrow \ (\exists x A(x) \cap \exists x B(x))),$$

where $A(x), B(x)$ are any formulas with a free variable x

The **inverse** implication **is not** a predicate tautology, i.e.

$$\not\models_p \ ((\exists x A(x) \cap \exists x B(x)) \Rightarrow \exists x\ (A(x) \cap B(x)))$$

Quantifiers Laws

To **prove** it we have to find an example of **concrete** formulas $A(x),\ B(x) \in \mathcal{F}$ and a model structure $\mathbf{M} = (U, I)$ with the interpretation I, such that **M** is **counter- model** for the formula

$$F:\ ((\exists x A(x) \cap \exists x B(x)) \Rightarrow \exists x\ (A(x) \cap B(x)))$$

We define the **counter - model** for F is as follows

Take $\mathbf{M} = (R, I)$ where R is the set of real numbers

Let $A(x),\ B(x)$ be **atomic** formulas $Q(x, c),\ \P(x, c)$

We define the interpretation I as $Q_I : >,\quad P_I : <,\quad c_I : 0$.

The formula F becomes an obviously **false** mathematical statement

$$F_I : ((\exists_{x \in R}\ x > 0 \cap \exists_{x \in R}\ x < 0) \Rightarrow \exists_{x \in R}\ (x > 0 \cap x < 0))$$

Quantifiers Laws

Distributivity of universal quantifier over disjunction holds only on **one direction**, namely the following is a predicate tautology for any formulas $A(x), B(x)$ with a free variable x.

$$\models_p \ ((\forall x A(x) \cup \forall x B(x)) \Rightarrow \forall x \ (A(x) \cup B(x))).$$

The inverse implication **is not** a predicate tautology, i.e.

$$\not\models_p \ (\forall x \ (A(x) \cup B(x)) \Rightarrow (\forall x A(x) \cup \forall x B(x)))$$

Quantifiers Laws

To **prove** it we have to find an example of **concrete** formulas $A(x),\ B(x) \in \mathcal{F}$ and a model structure $\mathbf{M} = (U, I)$ that is **counter- model** for the formula

$$F:\ (\forall x\,(A(x) \cup B(x)) \Rightarrow (\forall x A(x) \cup \forall x B(x)))$$

We take $\mathbf{M} = (R, I)$ where R is the set of real numbers, and $A(x),\ B(x)$ are **atomic** formulas $Q(x, c),\ \ R(x, c)$

We define $Q_I :\geq$ and $R_I :<,\ c_I : 0$

The formula F becomes an obviously **false** mathematical statement

$$F_I : (\forall_{x \in R}\ (x \geq 0 \cup x < 0) \Rightarrow (\forall_{x \in R}\ x \geq 0 \cup \forall_{x \in R}\ x < 0))$$

Logical Equivalence

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

Remember that $\equiv$ is not a new logical connective

This is a very useful symbol

It says that two formulas always have the same logical value

It can be used in the same way we the equality symbol $=$

Logical Equivalence

We formally define the **logical equivalence** as follows

Definition

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

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

We have also a similar definition for the propositional language and propositional tautology

Equational Laws for Quantifiers

De Morgan

For any formula $A(x) \in \mathcal{F}$ with a free variable x,

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

Definability

For any formula $A(x) \in \mathcal{F}$ with a free variable x,

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

Equational Laws for Quantifiers

Renaming the Variables

Let $A(x)$ be any formula with a free variable x

and let y be a variable that **does not occur** in A(x).

Let $A(x/y)$ be a result of **replacement** of each occurrence of x by y, then the following holds.

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

Alternations of Quantifiers

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

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.

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

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

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

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

Equational Laws for Quantifiers

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.

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

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

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

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

Equational Laws for Quantifiers

Distributivity Laws

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

Distributivity of universal quantifier over conjunction.

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

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 for Quantifiers

We also define the notion of logical equivalence $\equiv$ for the formulas of the **propositional language** and its semantics

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

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

Moreover, we prove that any substitution of **propositional tautology** by a formulas of the predicate language is a **predicate tautology**

The same holds for the logical equivalence

Equational Laws for Quantifiers

In particular, we transform the **propositional tautologies** into the following corresponding predicate equivalences.

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

$$(A \Rightarrow B) \equiv (\neg A \cup B),$$

$$(A \Rightarrow B) \equiv (\neg A \cup B)$$

We use them to prove the following De Morgan Laws for **restricted quantifiers**.

Equational Laws for Quantifiers

Restricted De Morgan

For any formulas $A(x), B(x) \in \mathcal{F}$ with a free variable x,

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

Here is a poof of first equality. The proof of the second one is similar and is left as an exercise.

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

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

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

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