# Introduction to Mathematical Logic, Edition 2017

**2 authors**, including:

Karlis Podnieks
University of Latvia
**78** PUBLICATIONS   **299** CITATIONS

SEE PROFILE

Some of the authors of this publication are also working on these related projects:

Project   Mathematical Logic and Foundations of Mathematics View project

Project   Philosophy of Modeling in 1870s: a Tribute to Hans Vaihinger View project

# Introduction to Mathematical Logic

Hyper-textbook for students

by **Vilnis Detlovs**, Dr. math.,
and **Karlis Podnieks**, Dr. math.

**University of Latvia**

Sections 1, 2, 3 of this book represent an extended translation of the corresponding chapters of the book: **V. Detlovs, Elements of Mathematical Logic,** Riga, University of Latvia, 1964, 252 pp. (in Latvian). With kind permission of Dr. Detlovs.

**Vilnis Detlovs. Memorial Page**

**In preparation** – forever (however, since 2000, used successfully in a real logic course for computer science students).

This hyper-textbook contains links to:
Wikipedia, the free encyclopedia;
MacTutor History of Mathematics archive
of the University of St Andrews;
MathWorld of Wolfram Research.

# Table of Contents

# References

**Hilbert D., Bernays P. [1934]** Grundlagen der Mathematik. Vol. I, Berlin, 1934, 471 pp. (Russian translation available)

**Kleene S.C. [1952]** Introduction to Metamathematics. Van Nostrand, 1952 (Russian translation available)

**Kleene S.C. [1967]** Mathematical Logic. John Wiley & Sons, 1967 (Russian translation available)

**Mendelson E. [1997]** Introduction to Mathematical Logic. Fourth Edition. International Thomson Publishing, 1997, 440 pp. (Russian translation available)

**Podnieks K. [1997]** What is Mathematics: Gödel's Theorem and Around. 1997-2015 (available online, Russian version available).

# 1. Introduction. What Is Logic, Really?

> **WARNING!** In this book,
>
> **predicate language** is used as a synonym of **first order language**,
>
> **formal theory** – as a synonym of **formal system, deductive system,**
>
> **constructive logic** – as a synonym of **intuitionistic logic,**
>
> **algorithmically solvable** – as a synonym of **recursively solvable,**
>
> **algorithmically enumerable** – as a synonym of **recursively enumerable.**

What is logic?

See also Factasia Logic by Roger Bishop Jones.

In a sense, logic represents the most general **means of reasoning** used by people and computers.

Why are means of reasoning important? Because any body of data may contain not only facts visible directly. For example, assume the following data: the date of birth of some person X is January 1, 2000, and yesterday, September 14, 2010 some person Y killed some person Z. Then, most likely, X did not kill Z. This conclusion is not represented in our data directly, but can be derived from it by using some means of reasoning – **axioms** ("background knowledge") and **rules of inference**. For example, one may use the following statement as an axiom: "Most likely, a person of age 10 can´t kill anybody".

There may be means of reasoning of different levels of generality, and of different ranges of applicability. The above "killer axiom" represents the lowest level – it is a very specific statement. But one can use laws of physics to derive conclusions from his/her data. Theories of physics, chemistry, biology etc. represent a more general level of means of reasoning. But can there be means of reasoning applicable in almost every situation? This – the most general – level of means of reasoning is usually regarded as **logic**.

Is logic absolute (i.e. unique, predestined) or relative (i.e. there is more than one kind of logic)? In modern times, an absolutist position is somewhat inconvenient – you must defend your "absolute" concept of logic against heretics and dissidents, but very little can be done to exterminate these people. They may freely publish their concepts on the Internet.

So let us better adopt the relativist position, and define logic(s) as **any common framework for building theories**. For example, the so-called absolute geometry can be viewed as a common logic for both the Euclidean

and non-Euclidean geometry. Group axioms serve as a common logic for theories investigating mathematical structures that are subtypes of groups. And, if you decide to rebuild all mathematical theories on your favorite set theory, then you can view set theory as your logic.

Can there be a common logic for the entire mathematics? To avoid the absolutist approach let us appreciate all the existing concepts of mathematics – classical (traditional), constructivist (intuitionist), New Foundations etc. Of course, enthusiasts of each of these concepts must propose some specific common framework for building mathematical theories, i.e. some specific kind of logic. And they do.

Can set theory (for example, currently, the most popular version of it – Zermelo-Fraenkel's set theory) be viewed as a common logic for the classical (traditional) mathematics? You may think so, if you do not wish to distinguish between the first order notion of natural numbers (i.e. discrete mathematics) and the second order notion (i.e. "continuous" mathematics based on set theory or a subset of it). Or, if you do not wish to investigate in parallel the classical and the constructivist (intuitionist) versions of some theories.

## 1.1. Total Formalization is Possible!

Gottlob Frege (1848-1925)
Charles S. Peirce (1839-1914)
Bertrand Russell (1872-1970)
David Hilbert (1862-1943)

How far can we proceed with the mathematical rigor – with the axiomatization of some theory? Complete elimination of intuition, i.e. full reduction of all proofs to a list of axioms and rules of inference, is this really possible? The work by Frege, Peirce, Russell, Hilbert and their colleagues and followers showed how this can be achieved even with the most complicated mathematical theories. All mathematical theories were indeed reduced to systems of axioms and rules of inference without any admixture of sophisticated human skills, intuitions etc. Today, the logical techniques developed by these brilliant people allow ultimate axiomatization of any theory that is based on a stable, self-consistent system of principles (i.e. of any mathematical theory).

Total axiomatization is called **formalization**. Today, we have another, very fundamental reason to formalize our mathematical and non-mathematical knowledge: in this way we can transmit this knowledge to **computers:** creating knowledge bases and systems of automatical reasoning.

What do they look like – such "100% rigorous" theories? They are called **formal theories** (the terms "formal systems" and "deductive systems" also are used) emphasizing that no step of reasoning can be done without a reference to an exactly formulated list of axioms and rules of inference. Even the most

"self-evident" logical principles (like, "if *A* implies *B*, and *B* implies *C*, then *A* implies *C*") must be either formulated in the list of axioms and rules explicitly, or derived from it.

What is, in fact, a mathematical theory? It is an **"engine" generating theorems**. Then, a formal theory must be an "engine" generating theorems without involving of human skills, intuitions etc., i.e. by means of a precisely defined algorithm, or a computer program.

The first distinctive feature of a formal theory is a **precisely defined** ("formal") **language** used to express its **propositions**. "Precisely defined" means here that there is an algorithm allowing to determine, is a given character string a correct proposition, or not.

The second distinctive feature of a formal theory is a precisely defined ("formal") notion of **proof**. Each proof proves some proposition, that is called (after being proved) a **theorem**. Thus, theorems are a subset of propositions.

It may seem surprising to a mathematician, but the most general exact definition of the "formal proof" involves neither axioms, nor inference rules. Neither "self-evident" axioms, nor "plausible" rules of inference are distinctive features of the "formality". Speaking strictly, "self-evident" is synonymous to "accepted without argumentation". Hence, axioms and/or rules of inference may be "good, or bad", "true, or false", and so may be the theorems obtained by means of them. The only definitely verifiable thing is here the fact that some theorem has been, indeed, **proved** by using some set of axioms, and by means of some set of inference rules.

Thus, the second distinctive feature of "formality" is the possibility to **verify the correctness of proofs** mechanically, i.e. without involving of human skills, intuitions etc. This can be best formulated by using the (since 1936 – precisely defined) notion of algorithm (a "mechanically applicable computation procedure"):

**A theory *T* is called a formal theory, if and only if there is an algorithm allowing to verify, is a given text a correct proof via principles of *T*, or not.** If somebody is going to publish a "mathematical text" calling it "*proof of a theorem in theory T*", then we must be able to verify it mechanically whether the text in question is really a correct proof according to the standards of proving accepted in theory *T*. Thus, in a formal theory, the standards of reasoning should be defined precisely enough to enable verification of proofs by means of a precisely defined algorithm, or a computer program. (Note that we are discussing here **verification of ready proofs**, and not the much more difficult problem – is some proposition provable in *T* or not, see Exercise 1.1.5 below and the text after it).

Axioms and rules of inference represent only one (but the most popular!) of

the possible techniques of formalization.

As an unpractical example of a formal theory let us consider the *game of chess*, let us call this "theory" *CHESS*. Let 's define as **propositions** of *CHESS* all the possible positions – i.e. allocations of some of the pieces (kings included) on a chessboard – plus the flag: "whites to move" or "blacks to move". Thus, the set of all the possible positions represents the **language** of *CHESS*. The only **axiom** of *CHESS* is the initial position, and the **rules of inference** – the rules of the game. Rules allow passing from some propositions of *CHESS* to some other ones. Starting with the axiom and iterating moves allowed by the rules we obtain **theorems** of *CHESS*. Thus, theorems of *CHESS* are defined as all the possible positions (i.e. propositions of *CHESS*) that can be obtained from the initial position (the axiom of *CHESS*) by moving pieces according to the rules of the game (i.e. by using the inference rules of *CHESS*).

**Exercise 1.1.1** (optional). Could you provide an **unprovable** proposition of *CHESS*? Try proving that your proposition is unprovable, indeed, i.e. that your position cannot be obtained from the initial position by moving pieces according to the rules of the game.

(We will see later that in logic, "negative" proving, i.e. proving that something cannot be proved, is sometimes a more complicated task than "positive" proving.)

Why is *CHESS* called a formal theory? When somebody offers a "mathematical text" $P$ as a proof of a theorem $A$ in *CHESS*, this means that $P$ is a record of some chess-game stopped in the position $A$. Checking the correctness of such "proofs" is a boring, but an easy task. The rules of the game are formulated precisely enough – we could write a computer program that will execute the task.

**Exercise 1.1.2** (optional). Try estimating the size of this program in some programming language.

Our second example of a formal theory is only a bit more serious. It was proposed by Paul Lorenzen, so let us call this theory $L$. **Propositions** of $L$ are all the possible "words" made of letters *a, b,* for example: *a, b, aa, aba, baab.* Thus, the set of all these "words" is the **language** of $L$. The only **axiom** of $L$ is the word *a*, and $L$ has two **rules of inference**: $X \vdash Xb$, and $X \vdash aXa$. This means that (in $L$) from a proposition $X$ we can infer immediately the propositions $Xb$ and $aXa$. For example, the proposition *aababb* is a **theorem** of $L$:

$$a \vdash ab \vdash aaba \vdash aabab \vdash aababb$$
$$\text{rule1} \quad \text{rule2} \quad \text{rule1} \quad \text{rule1}$$

This fact is expressed usually as $L \vdash aababb$ ( "L proves *aababb*", $\vdash$ being a "fallen T").

**Exercise 1.1.3.** a) Verify that *L* is a formal theory. (Hint: describe an algorithm allowing to determine, is a sequence of propositions of *L* a correct proof, or not.)

b) (P. Lorenzen) Verify the following property of theorems of L: for any *X*, if $L \vdash X$, then $L \vdash aaX$.

One of the most important properties of formal theories is given in the following

**Exercise 1.1.4.** Show that the **set of all theorems of a formal theory is algorithmically enumerable**, i.e. show that, for any formal theory T, an algorithm $A_T$ can be defined that prints out on an (endless) paper tape all theorems of this theory (and nothing else). (Hint: we will call *T* a formal theory, if and only if we can present an algorithm for checking texts as correct proofs via principles of reasoning of *T*. Thus, assume, you have 4 functions: *GenerateFirstText()* – returns *Text*, *GenerateNextText()* – returns *Text*, *IsCorrectProof(Text)* – returns *true* or *false*, *ExtractTheorem(Text)* – returns *Text*, and you must implement the functions *GenerateFirstTheorem()* – returns *Text*, *GenerateNextTheorem()* – returns *Text*).

Unfortunately, **such algorithms and programs cannot solve the problem that the mathematicians are mainly interested in: is a given proposition *A* provable in *T* or not?** When, executing the algorithm $A_T$, we see our proposition *A* printed out, this means that *A* is provable in *T*. Still, in general, until that moment we cannot know in advance whether *A* will be printed out some time later or it will not be printed at all.

**Note.** According to the official terminology, algorithmically enumerable sets are called "recursively enumerable sets", in some texts – also "listable sets".

**Exercise 1.1.5.** a) Describe an algorithm determining whether a proposition of *L* is a theorem or not.

b) (optional) Could you imagine such an algorithm for *CHESS*? Of course, you can, yet... Thus you see that even, having a relatively simple algorithm for *checking the correctness* of proofs, the problem of *provability* can be a very complicated one.

*T* is called a **solvable theory** (more precisely – **algorithmically solvable theory**), if and only if there is an algorithm allowing to check whether some proposition is *provable* by using the principles of *T* or not. In the Exercise 1.1.5(a) you proved that *L* is a solvable theory. Still, in the Exercise 1.1.5(b) you established that it is hard to state whether *CHESS* is a "feasibly solvable"

theory or not. **Determining the provability of propositions is a much more complicated task than checking the correctness of ready proofs.** It can be proved that **most mathematical theories are unsolvable**, the elementary (first order) arithmetic of natural numbers and set theory included (see, for example, Mendelson [1997], or Podnieks [1997], Section 6.3). I.e. there is no algorithm allowing to determine, is some arithmetical proposition provable from the axioms of arithmetic, or not.

**Note.** According to the official terminology, algorithmically solvable sets are called "recursive sets".

Normally, mathematical theories contain the negation symbol **not**. In such theories solving the problem stated in a proposition *A* means proving either *A,* or proving **not***A* ("disproving *A*", "refuting *A*"). We can try to solve the problem by using the enumeration algorithm of the Exercise 1.1.4: let us wait until *A* or **not***A* is printed. If *A* and **not***A* will be printed both, this will mean that *T* is an **inconsistent theory** (i.e. using principles of *T* one can prove some proposition and its negation). In general, we have here 4 possibilities:

a) *A* will be printed, but **not***A* will not (then the problem *A* has a positive solution),

b) **not***A* will be printed, but *A* will not (then the problem *A* has a negative solution),

c) *A* and **not***A* will be printed both (then *T* is an inconsistent theory),

d) neither *A*, nor **not***A* will be printed.

In the case d) we may be waiting forever, yet nothing interesting will happen: using the principles of *T* one can neither prove nor disprove the proposition *A,* and for this reason such a theory is called an **incomplete theory**. The famous incompleteness theorem proved by Kurt Gödel in 1930 says that **most mathematical theories are either inconsistent or incomplete** (see Mendelson [1997] or Podnieks [1997], Section 6.1).

**Exercise 1.1.6.** Show that any (simultaneously) consistent and complete formal theory is solvable. (Hint: use the algorithm of the Exercise 1.1.4, i.e. assume that you have the functions *GenerateFirstTheorem()* − returns *Text*, *GenerateNextTheorem()* − returns *Text*, and implement the function *IsProvable(Text)* – returns *true* or *false*). Where the consistency and completeness come in?

**Exercise 1.1.7** (optional). a) Verify that "fully axiomatic theories" are formal theories in the sense of the above general definition. (Hint: assume, that you have the following functions: *GenerateFirstText()* − returns *Text*, *GenerateNextText()* − returns *Text*, *IsPropositon(Text)* − returns *true* or *false*, *IsAxiom(Proposition)* − returns *true* or *false*, there is a finite list of inference rule names: $\{R_1, ..., R_n\}$, function *Apply(RuleName, ListOfPropositions)* − returns *Proposition* or *false*, and you must implement the functions *IsCorrectProof(ListOf*

*Propositions) −* returns *true* or *false*, *ExtractTheorem(Proof) −* returns *Proposition*).

b) (for smart students) What, if, instead of $\{R_1, ..., R_n\}$, we would have an *infinite* list of inference rules, i.e. functions *GenerateFirstRule()*, *GenerateNextRule()* returning *RuleName*?

## 1.2. Predicate Languages

### History

For a short overview of the history, see Quantification.

See also:

Aristotle (384-322 BC) – in a sense, the "first logician", "... was not primarily a mathematician but made important contributions by systematizing deductive logic." (according to MacTutor History of Mathematics archive).

Gottlob Frege: "In 1879 Frege published his first major work *Begriffsschrift, eine der arithmetischen nachgebildete Formelsprache des reinen Denkens* (Conceptual notation, a formal language modelled on that of arithmetic, for pure thought). A.George and R Heck write: *... In effect, it constitutes perhaps the greatest single contribution to logic ever made and it was, in any event, the most important advance since* Aristotle. *...* In this work Frege presented for the first time what we would recognise today as a logical system with negation, implication, universal quantification, essentially the idea of truth tables etc." (according to MacTutor History of Mathematics archive).

**Hilary Putnam.** Peirce the Logician. *Historia Mathematica*, Vol. 9, 1982, pp. 290-301 (an online excerpt available, published by John F. Sowa).

**Richard Beatty.** Peirce's development of quantifiers and of predicate logic. *Notre Dame J. Formal Logic*, Vol. 10, N 1 (1969), pp. 64-76.

**Geraldine Brady**. From Peirce to Skolem. A Neglected Chapter in the History of Logic. Elsevier Science: North-Holland, 2000, 2000, 625 pp.

When trying to formalize some piece of our (until now – informal) knowledge, how should we proceed? We have an informal vision of some domain consisting of "objects". When speaking about it, we are uttering various *propositions*, and some of these propositions are regarded as "true" statements about the domain.

"When trying to formalize some piece of our knowledge..." - why should we? Today, we have another, very fundamental reason to formalize our knowledge: in this way we can transmit this knowledge to computers.

Thus, our first formalization task must be defining of some *formal language*, allowing to put all our propositions about the domain in a uniform and precise way.

After this, we can start considering propositions that we are regarding as

"true" statements about the domain. There may be an infinity of such statements, hence, we can't put down all of them, so we must organize them somehow. Some minimum of the statements we will simply declare as *axioms*, the other ones we will try to derive from the axioms by using some precisely defined *rules of inference*.

As the result, we will obtain a *formal theory* (in the sense of the previous Section).

Formulation of axioms (in knowledge bases they are called *facts* and *rules*) and rules of inference (i.e. logic) is an absolutely necessary next step, if we wish to transmit our knowledge to computers. From scratch, computers "do not know" even the most trivial things about, for example, the human society, such as "persons cannot be parents of themselves". Or, about natural numbers: we must submit, for example, $x+0=x$ as an axiom to our computers, if we wish teach to them proving of mathematical theorems.

In mathematics and computer science, the most common approach to the first phase of formalization is using of the so-called **predicate languages**, first introduced by G. Frege and C. S. Peirce.

(In many textbooks, they are called **first order languages**, see below the warning about second order languages.)

Usually, linguists analyze the sentence "John loves Britney" as follows: *John* – subject, *loves* – predicate, *Britney* – object. The main idea of predicate languages is as follows: instead, let us write *loves(John, Britney)*, where *loves(x, y)* is a two-argument predicate, and *John*, and *Britney* both are objects. Following this way, we could write *=(x, y)* instead of *x=y*. This approach – reducing of the human language sentences to variables, constants, functions, predicates and **quantifiers** (see below), appears to be flexible enough, and it is much more uniform when compared to the variety of constructs used in the natural human languages. A unified approach is much easier to use for communication with computers.

Another example: "Britney works for BMI as a programmer". In a predicate language, we must introduce a 3-argument predicate "x works for y as z", or *works(x, y, z)*. Then, we may put the above fact as: *works(Britney, BMI, Programmer)*.


**Language primitives**

Thus, the informal vision behind the notion of predicate languages is centered on the so-called "domain" – a (non-empty?) collection of "objects", their "properties" and the "relations" between them, that we wish to "describe" (or "define"?) by using the language. This vision serves as a guide in defining the language precisely, and further – when selecting axioms and rules of inference.
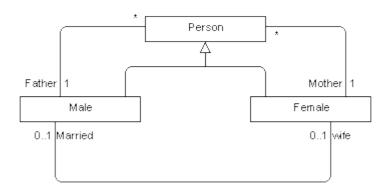
## Object variables

Thus, the first kind of language elements we will need are **object variables** (sometimes called also individual variables, or simply, variables). We need an unlimited number of them):

$$x, y, z, x_1, y_1, z_1, ...$$

The above-mentioned "domain" is the intended "range" of all these variables.

**Examples.** 1) Building a language that should describe the "domain of people", we must start by introducing "variables for people": x denotes an arbitrary person.

2) Building the language of the so-called **first order arithmetic**, we are thinking about "all natural numbers" as the range of variables: 0, 1, 2, 3, 4, ... (x denotes an arbitrary natural number).

3) Building the language of **set theory**, we think about "all sets" as the range of variables: x denotes an arbitrary set.



"*Domain of people" represented as [UML class diagram](UML class diagram)*

**Note.** Since our screens and printers allow only a limited number of pixels per inch, in principle, we should generate variable names by using a finite set of characters, for example, by using a single letter x:

$$x, xx, xxx, xxxx, xxxxx, ...$$

## Object constants

The next possibility we may wish to have in our language are the so-called **object constants** (sometimes called individual constants, constant letters, or simply, constants) – names or symbols denoting some specific "objects" of our "domain".

**Examples**. 1) In our "language for people" we may introduce constants denoting particular people: John, Britney etc.

2) In the language of first order arithmetic, we may wish to introduce two constants – 0 and 1 to denote "zero" and "one" – two natural numbers having specific properties.

3) In the language of set theory, we could introduce a constant denoting the empty set, but there is a way to do without it as well (for details, Podnieks [1997], Section 2.3).

## Function constants

In some languages we may need also the so-called **function constants** (sometimes called function letters) – names or symbols denoting specific functions, i.e. mappings between "objects" of our "domain", or operations on these objects.

**Examples**. 1) In our "language for people" we will not use function constants.

2) In the language of first order arithmetic, we will introduce only two function constants "+" and "*" denoting the usual addition and multiplication of natural numbers, i.e. the two-argument functions x+y and x*y.

3) In the language of set theory, we could introduce function constants denoting set intersections $x \cap y$, unions $x \cup y$, set differences x–y, power sets P(x) etc., but there is a way to do without these symbols as well (for details, Podnieks [1997], Section 2.3).

In mathematics, normally, we are writing *f(x, y)* to denote the value of the function *f* for the argument values x, y. This (the so-called "prefix" notation) is a uniform way suitable for functions having any number of arguments: *f(x), g(x, y), h(x, y, z)* etc. In our everyday mathematical practice some of the two-argument functions (in fact, operations) are represented by using the more convenient "infix" notation (x+y, x*y instead of the uniform +(x, y), *(x, y), etc.).

**Note**. In a sense, object constants can be viewed as a special case of function constants – an object constant is a "zero-argument function".

## Predicate constants

The last (but the most important!) kind of primitives we need in our language are the so-called **predicate constants** (sometimes called predicate letters) – names or symbols denoting specific **properties** (of) or **relations** between "objects" of our "domain".

**Note**. Using "predicate" as the unifying term for "property" and "relation" may seem somewhat unusual. But some kind of such unifying term is necessary. Properties are, in fact, unary (i.e. one-argument) "predicates", for example, "x is red". Relations are, two- or more-argument "predicates", for example, "x is better than y", or "x sends y to z".

**Examples**. 1) In our "language for people" we will use the following predicate constants (see the class diagram above):

*Male(x)* ‒ means "x is a male";

*Female(x)* ‒ means "x is a female";

*Mother(x, y)* ‒ means "x is mother of y";

*Father(x, y)* ‒ means "x is father of y";

*Married(x, y)* ‒ means "x and y are married, y being wife";

x=y ‒ means "x an y are the same person".

The first two constants represent, in fact, "properties" (or, "classes") of our objects. The other 4 constants represents "relations" between our objects. The term "predicate" is used to include both versions. We do not introduce *Person(x)* as a predicate because our domains consists of persons only.

2) It may seem strange to non-mathematicians, yet the most popular relation of objects used in most mathematical theories, is **equality** (or identity). Still, this is not strange for mathematicians. We can select an object x in our "domain" by using a very specific combination of properties and relations of it, and then – select another object y – by using a different combination. And after this (sometimes it may take many years to do) we prove that x=y, i.e. that these two different combinations of properties and relations are possessed by a single object. Many of discoveries in mathematics could be reduced to this form.

In the language of first order arithmetic, equality "=" is the only necessary predicate constant. Other"basic" relations must be reduced to equality. For example, the relation x<y for natural numbers x, y can be reduced to equality by using the addition function and the formula $\exists z(x+z+1=y)$.

3) In the language of set theory a specific predicate constant "in" denotes the set membership relation: "x in y" means "x is a member of y". The equality predicate x=y also will be used – it means "the sets x an y possess the same members".

The uniform way of representation suitable for predicates having any number of arguments is again the "prefix" notation: *p(x), q(x, y), r(x, y, z)* etc. In the real mathematical practice, some of the two-argument predicates are represented by using the "infix" notation (for example, x=y instead of the uniform =(x, y), etc.).

**Zero-argument predicate constants?** In an interpretation (see below), each such predicate must become either "true", or "false". Hence, paradoxically, zero-argument predicate constants behave like "propositional variables" – they represent assertions that do not possess a meaning, but possess a "truth value".

## Summary of language primitives

Thus, the specification of a predicate language includes the following **primitives**:

1) A countable set of object variable names (you may generate these names, for example, by using a single letter "x": x, xx, xxx, xxxx, ...).

2) An empty, finite, or countable set of object constants.

3) An empty, finite, or countable set of function constants. To each function constant a fixed argument number must be assigned.

4) A finite, or countable set of predicate constants. To each predicate constant a fixed argument number must be assigned.

Different sets of primitives yield different predicate languages.

**Examples**. 1) Our "language for people" is based on: a) object variables x, y, z, ...; b) object constants: John, Britney, ...; c) function constants: none; d) predicate constants: *Male(x), Female(x), Mother(x, y), Father(x, y), Married(x, y), x=y.*

2) The language of first order arithmetic is based on: a) object variables x, y, z, ...; b) object constants: 0, 1; c) function constants: x+y, x*y; d) predicate constant: x=y.

3) The language of set theory is based on: a) object variables x, y, z, ...; b) object constants: none; c) function constants: none; d) predicate constants: x in y, x=y.

The remaining part of the language definition is common for all predicate languages.

### Terms and formulas

By using the language primitives, we can build terms, atomic formulas and (compound) formulas.

**Terms** are expressions used to denote objects and functions:

a) Object variables and object constants (if any), are terms.

b) If f is a k-argument function constant, and $t_1$, ..., $t_k$ are terms, then the string $f(t_1, ..., t_k)$ is a term.

c) There are no other terms.

**Examples**. 1) In our "language for people" only variables x, y, z, ..., and object constants *John, Britney, ...* are terms.

2) In the language of first order arithmetic, for addition and multiplication the "infix" notation is used: if $t_1$, $t_2$ are terms, then the $(t_1+t_2)$ and $(t_1*t_2)$ are terms. Of course, the object constants 0, 1 and variables x, y, z, ... are terms. Examples of more complicated terms: (x+y), ((1+1)*(1+1)), (((1+1)*x)+1).

3) In the language of set theory, variables x, y, z, ... are the only kind of terms.

If a term does not contain variable names, then it denotes an "object" of our "domain" (for example, ((1+1)+1) denotes a specific natural number – the number 3). If a term contains variables, then it denotes a function. For example, (((x*x)+(y*y))+1) denotes the function $x^2+y^2+1$.

**Warning!** Note that the language of first order arithmetic **does not** contain a function constant denoting the **exponentiation** $x^y$, thus, for example, we must

write x*x instead of $x^{1+1}$.

Of course, the key element of our efforts in describing "objects", their properties and relations, will be assertions, for example, the commutative law in arithmetic: ((x+y)=(y+x)). In predicate languages, assertions are called **formulas** (or, sometimes, well formed formulas – wff-s, sentences, or statements).

**Atomic formulas** (in some other textbooks: elementary formulas, prime formulas) are defined as follows:

a) If p is a k-argument predicate constant, and $t_1$, ..., $t_k$ are terms, then the string $p(t_1, ..., t_k)$ is an atomic formula.

b) There are no other atomic formulas.

For the equality symbol, the "infix" notation is used: if $t_1$, $t_2$ are terms, then $(t_1=t_2)$ is an atomic formula.

**Examples**. 1) In our "language for people", the following are examples of atomic formulas: *Male(x), Female(Britney), Male(Britney)* (not all formulas that are well formed, must be true!), *Father(x, Britney), Mother(Britney, John), Married(x, y)*.

2) Summary of the atomic formulas of the language of first order arithmetic: a) constants 0 and 1, and all variables are terms; b) if $t_1$ and $t_2$ are terms, then $(t_1+t_2)$ and $(t_1*t_2)$ also are terms; c) atomic formulas are built only as $(t_1=t_2)$, where $t_1$ and $t_2$ are terms.

3) In the language of set theory, there are only two kinds of atomic formulas: $x \in y$ , and x=y (where x and y are arbitrary variables).

In the language of first order arithmetic, even by using the only available predicate constant "=" atomic formulas can express a lot of clever things, for example,

$$((x+0)=x); ((x+y)=(y+x)); ((x+(y+z))=((x+y)+z));$$
$$((x*0)=0); ((x*1)=x); ((x*y)=(y*x)); ((x*(y*z))=((x*y)*z));$$
$$(((x+y)*z)=((x*z)+(y*z))).$$

**Exercise 1.2.1.** As the next step, translate the following assertions into the language of first order arithmetic (do not use abbreviations!): 2*2=4, 2*2=5, $(x+y)^2 = x^2+2xy+y^2$.

**(Compound) formulas**

**The following definition is the common part of all predicate languages. Each language is specific only by its set of language primitives.**

To write more complicated assertions, we need compound formulas, built of atomic formulas by using a fixed set of **propositional connectives** and **quantifiers** (an invention due to G. Frege and C. S. Peirce). In this book, we

will use the following set:

Implication symbol: B→C means "if B, then C", or "B implies C", or "C follows from B".

Conjunction symbol, $B \land C$ means "B and C".

Disjunction symbol, $B \lor C$ means "B, or C, or both", i.e. the so-called non-exclusive "or".

**Warning!** Once again, disjunction means the so-called **non-exclusive "or"**. If B and C both are true, then $B \lor C$ is true as well. The so-called **exclusive "or"** (programmers would write it as $B \, xor \, C$ ) is used to denote "either B, or C, but not both".

Negation symbol, $\neg B$ means "not B".

Universal quantifier, $\forall x \, B$ means "for all x: B".

Existential quantifier, $\exists x \, B$ means "there is x such that B".

The widely used equivalence connective ↔ can be derived in the following way: B↔C stands for $((B \to C) \land (C \to B))$ . If you like using the so-called exclusive "or" ("either B, or C, but not both"), you can define $B \, xor \, C$ as $\neg (B \leftrightarrow C)$ .

**Warning!** For programmers, conjunction, disjunction and negation are familiar "logical operations" – unlike the implication that is not used in "normal" programming languages. In programming, the so-called IF-statements, when compared to logic, mean a different thing: in the statement IF x=y THEN z:=17, the condition, x=y is, indeed, a formula, but the "consequence" z:=17 is not a formula – it is an executable statement. In logic, in B→C ("if B, then C"), B and C both are formulas.

We define the notion of **formula** of our predicate language as follows:

a) Atomic formulas are formulas.

b) If B and C are formulas, then $(B \to C), (B \land C), (B \lor C)$ , and $(\neg B)$ also are formulas (B and C are called sub-formulas).

c) If B is a formula, and x is an object variable, then $(\forall x \, B)$ and $(\exists x \, B)$ also are formulas (B is called a sub-formula).

d) (If you like so,) there are no other formulas.

See also:

Notes on Logic Notation on the Web by Peter Suber.

**Warning about omitting parentheses!** To make formulas easier for human reading, usually, some of the formally necessary parentheses are omitted according to the so-called *priority rules*. For example,

$B \lor C \to D \land F$  means  $(B \lor C) \to (D \land F)$ , and **not**  $(B \lor (C \to D) \land F)$ .

Thus, implication has lower priority than disjunction and conjunction. For full treatment of priority rules see the section "Omitting Parentheses" below.

### Knowledge representation by means of predicate languages

Do not be surprised by the trivial character of the most of the assertions expressed in the formulas below. Even such trivial assertions must be submitted to computers, if we wish them make conclusions and answer questions about people, natural numbers, or sets.

**Examples**. 1) In our "language for people", the following are examples of compound formulas:

| | |
|---|---|
| $((Father(x,y)) \lor (Mother(x,y)))$ | "x is a parent of y" |
| $(\forall x(\forall y((Father(x,y)) \to (Male(x)))))$ | "fathers are males" – could serve as an **axiom**! Looks **somewhat artificial**, but represents the most natural way of saying that "fathers are males" in a predicate language! |
| $(\forall x(\forall y((Mother(x,y)) \to (\neg Male(x)))))$ | "mothers are not males" – could be **derived** from the axioms! |
| $(\forall x(\exists y(Mother(y,x))))$ | "each x has some y as a mother" – could serve as an **axiom**! |
| $(\forall x(Male(x) \lor Female(x)))$ | What does it mean? It could serve as an **axiom**! |
| $(\forall x \neg (Male(x) \land Female(x)))$ | What does it mean? It could serve as an **axiom**! |

| |
|---|
| $\exists z(Father(x,z) \land (Father(z,y) \lor Mother(z,y)))$   "x is a grandfather of y" |
| The existential quantifier looks **somewhat artificial** here, but this formula represents the most natural way of saying such things in a predicate language! |

| |
|---|
| $\forall x(\forall y(\forall z((Mother(x,z) \land Mother(y,z)) \to (x=y))))$ |
| $\forall x(\forall y(\forall z((Father(x,z) \land Father(y,z)) \to (x=y))))$ |
| What does this mean? These formulas could serve as **axioms**! Looks **somewhat artificial**, but represents the most natural way of saying "no more than one" in a predicate language! |

2) Some simple examples of compound formulas in the language of first order arithmetic:

**Warning!** Speaking strictly, predicate symbols "<", ">", "≤", "≥", "≠" etc. do not belong to

the language of first order arithmetic.

| | |
|---|---|
| (∃u(x=(u+u))) | "x is an even number" |
| (∃u(((x+u)+1)=y)) | "x is less than y", or, x<y |
| $(0 < y \wedge \exists u (x = (y * u)))$ | "x is divisible by y". Speaking strictly, x<y must be replaced by $\exists u(((x+u)+1)=y)$ . |
| $((1 < x) \wedge (\neg (\exists y (\exists z (((y < x) \wedge (z < x)) \wedge (x = (y * z)))))))$ <br><br> formula *prime(x)*, "x is a prime number", Speaking strictly, the 3 subformulas of the kind x<y should be replaced by their full version of the kind $\exists u((((x+u)+1)=y))$ . | |
| $\forall w (\exists x ((w < x) \wedge (prime(x))))$ | "There are infinitely many prime numbers" (one of the first mathematical theorems, VI century BC). Speaking strictly, w<x must be replaced by $\exists u((((w+u)+1)=x))$ , and *prime(x)* must be replaced by the above long formula. |
| $\forall x \forall y (0 < y \rightarrow \exists z \exists u (u < y \wedge x = y * z + u))$ | What does it mean? |

3) Some simple examples of compound formulas in the language of set theory:

| | |
|---|---|
| $(\exists y (y \in x))$ | "x is a non-empty set" |
| $(\forall z ((z \in x) \rightarrow (z \in y)))$ | "x is a subset of y", or x≤y |
| $((\forall z ((z \in x) \leftrightarrow (z \in y))) \rightarrow (x = y))$ | What does it mean? Will serve as an **axiom**! |
| $(\forall y (\forall z ((y \in x) \wedge (z \in x)) \rightarrow y = z))$ | "x contains zero or one member" |
| $(\forall u ((u \in x) \leftrightarrow ((u \in y) \vee (u \in z))))$ | "x is union of y and z", or x=yUz |

Of course, **having a predicate language is not enough** for expressing all of our knowledge formally, i.e. for communicating it to computers. Computers do not know in advance, for example, how to handle sexes. We must tell them how to handle these notions by introducing **axioms**. Thus, the above-mentioned formulas like as

$$\forall x (Male(x) \vee Female(x)) \text{ , or } \forall x \forall y (Father(x, y) \rightarrow (Male(x)))$$

will be absolutely necessary as **axioms**. As we will see later,

$$language + axioms + logic = theory,$$

i.e. in fact, **to formulate all of our knowledge formally, we must create**

**theories.**

**Exercise 1.2.2.** Translate the following assertions into our "language for people":

<div align="center">

"x is child of y";

"x is grand-mother of y"; "x is a grand-father"

"x is brother of x"; "x and y are sisters";

"x is <u>cousin</u> of y"; "x is <u>nephew</u> of y"; "x is <u>uncle</u> of y".

</div>

**Exercise 1.2.3.** Translate the following assertion to the language of first order arithmetic:

"x and y do not have common divisors" (note: 1 is not counted as a divisor!);

" $\sqrt{2}$ is not a rational number"; (Warning! $\neg \exists \, p \, \exists \, q (\sqrt{2} = \frac{p}{q})$ , and

$\exists \, x (x * x = 2)$ are not correct solutions. Why?)

**Exercise 1.2.4.** Imagine an alternative language of arithmetic that does not contain function constants + and *, but contains predicates *sum(x, y, z)* and *prod(x, y, z)* instead (meaning x+y=z and x*y=z correspondingly). Translate the following assertions into this language:

$$x + 0 = x \, ; x + y = y + x \, ; x + (y + 1) = (x + y) + 1 \, ; (x + y) * z = (x * z) + (y * z) \quad .$$

**Exercise 1.2.5**. Try inventing your own predicate language. Prepare and do your own Exercise 1.2.2 for it.

**Exercise 1.2.6** (optional). In computer science, currently, one the most popular means of knowledge representation are the so-called **UML class diagrams** and **OCL** (UML − <u>Unified Modeling Language</u>, OCL − <u>Object Constraint Language</u>). The above diagram representing our "domain of people" is an example. In our "language of people", put down as many **axioms** of the domain you can notice in the diagram. For example, "every person is either male, or female", "all fathers are males", "every person has exactly one mother", "a person can marry no more than one person" etc.

---

**Warning! Some typical errors!**

1a) Trying to say "for all x>2, F(x)", do not write $\forall \, x (x > 2 \wedge F(x))$ . This formula would imply that $\forall \, x (x > 2)$ – a silly conclusion! Indeed, how about x=1? The correct version: $\forall \, x (x > 2 \rightarrow F(x))$ .

1b) Trying to say "there is x>2, such that F(x)", do not write $\exists x (x > 2 \rightarrow F(x))$ . The formula under the quantifier is true for x=1, hence, the entire formula cannot guarantee that "there is x>2, such that F(x)". The correct version: $\exists x (x > 2 \wedge F(x))$

2) Some computer programmers do not like using the implication

---

connective →, trying to write formulas as conditions of IF- or WHILE-statements, i.e. by using conjunction, disjunction and negation only. This "approach" makes most logical tasks much harder than they really are! More than that − some people try saying, for example, "Persons are not Departments", as crazy as follows: $\forall x(Person(x) \wedge \neg Department(x))$ − instead of the correct version:

$$\forall x(Person(x) \rightarrow \neg Department(x)) \ .$$

3) Do not use abbreviations at this early stage of your logical studies. For example, do not write *(∀x>2)∃yF(x, y)* to say that "for all x that are >2, there is y such that F(x, y)". Instead, you should write *∀x(x>2→∃yF(x, y))*. Similarly, instead of *(∃a>2)∃bG(a, b)*, you should write $\exists a(a > 2 \wedge \exists b G(a, b)) \ .$

To say "there is one and only one x such that F(x)", you shouldn't write *∃!x F(x)* (at this stage of your studies), you should write, instead,

$$((\exists x F(x)) \wedge (\forall x_1(\forall x_2((F(x_1) \wedge F(x_2)) \rightarrow (x_1 = x_2))))) \ .$$

Of course, *∃!x F(x)* could serve as a "macro" for the above longer formula.

4) Predicates cannot be substituted for object variables. For example, having 3 predicate constants *working(x, y), Person(x), Department(y)*, do not try writing *working(Person(x), Department(y))* to say that "only persons are working, and only in departments". The correct version:

$$\forall x \forall y(working(x, y) \rightarrow Person(x) \wedge Department(y)) \ .$$

5) Trying to say "each person is working in some department", do not write

$$\forall x \exists y(Person(x) \wedge Department(y) \rightarrow working(x, y)) \ .$$

The correct version:

$$\forall x(Person(x) \rightarrow \exists y(Department(y) \wedge working(x, y))) \ .$$

Think it over: what is the difference?

## Many-sorted Languages

Maybe, you have to describe two or more kinds of "objects" that you do not wish to reduce to "sub-kinds" of one kind of "objects" (for example, integer numbers and character strings). Then you may need introducing for each of your "domains" a separate kind ("sort") of object variables. In this way you arrive to the so-called **many-sorted predicate languages**. In such languages: a) a finite number of sorts must be introduced; b) each variable and each object constant must be assigned to some sort; b) for each function constant, each of its arguments must be assigned to some some sort, and function values must be assigned to a (single) sort; c) for each predicate constant, each of its argument must be assigned to some sort. In many-

sorted predicate languages, the term and atomic formula definitions are somewhat more complicated: building of the term $f(t_1, ..., t_k)$ or the formula $p(t_1, ..., t_k)$ is allowed only, if the sort of the term $t_i$ values coincides with the sort of the i-th argument of f or p respectively. And the "meaning" of quantifiers depends on the sort of the variable used with them. For example, $\forall x$ means "for all values of x from the domain of the sort of x".

Theoretically, many-sorted languages can be reduced to one-sorted languages by introducing the corresponding predicates $Sort_i(x)$ ("the value of x belongs to the sort i"). Still, in some applications of logic (for example, in computer science) the many-sorted approach is usually more natural and more convenient. (See *Chapter 10. Many-Sorted First Order Logic*, by **Jean Gallier**.)

## Warning about second order languages!

In our definition of predicate languages only the following kinds of primitives were used: object variables, object constants, function constants and predicate constants. You may ask: how about **function variables** and **predicate variables**? For, you may wish to denote by r "an arbitrary property" of your "objects". Then, r(x) would mean "x possess the property r", and you would be able to say something about "all properties", for example, $\forall r \forall x \forall y (x = y \rightarrow r(x) \leftrightarrow r(y))$. In this way you would have arrived at a **second order language**! In such languages, function and predicate variables are allowed. But properties lead to sets of objects, for example, {x | r(x)} would mean the set of all objects that possess the property r. But, why should we stop at the properties of objects? How about "properties of sets of objects" etc.? As it was detected long ago, all kinds of sets can be fully treated only in **set theory**! Thus, instead of building your own second order language, you should better try applying your favorite ("first order") set theory. An unpleasant consequence: the existence of the (much less significant) notion of second order languages forces many people to call predicate languages "**first order languages**" − to emphasize that, in these languages, the only kind of variables allowed are object variables.

On the other hand, when trying to implement realistic **formal reasoning software**, then using of some second order constructs is, as a rule, more efficient than implementing of a pure first order reasoning. See, for example, *Notices of the AMS*, Special Issue on Formal Proof, Vol. 55, N 11, 2008 (available online).

For details, see: Second-order-logic. About **second order arithmetic** see Reverse Mathematics. About an almost (but not 100%) successful attempt to create a set theory "as simple as logic" (by Georg Cantor and Gottlob Frege) – see Podnieks [1997], Section 2.2.

## Omitting parentheses

Our formal definitions of terms and formulas lead to expressions containing many parentheses. Let us remind, for example, our formula expressing that "x is a prime number":

$$((1 < x) \wedge (\neg (\exists y (\exists z (((y < x) \wedge (z < x)) \wedge (x = (y * z)))))))) .$$

Such formulas are an easy reading for computers, yet inconvenient for human reading (and even more inconvenient – for putting them correctly). In the

usual mathematical practice (and in programming languages) we are allowed to improve the look of our formulas by omitting some of the parentheses − according to (some of) the following rules:

a) Omit the outermost parentheses, for example, we may write A→(B→C) instead of the formally correct (A→(B→C)). In this way we may improve the final look of our formulas. Still, if we wish to use such formulas as parts of more complicated formulas, we must restore the outermost parentheses, for example: (A→(B→C))→D.

b) We may write, for example, simply:

$$x+y+z+u, x*y*z*u,$$
$$A \wedge B \wedge C \wedge D, A \vee B \vee C \vee D,$$
$$\exists x \forall y \exists z \forall u F$$

instead of the more formal

$$((x+y)+z)+u, ((x*y)*z)*u,$$
$$((A \wedge B) \wedge C) \wedge D,$$
$$((A \vee B) \vee C) \vee D$$
$$\exists x(\forall y(\exists z(\forall u(F)))).$$

In this way we can simplify the above expression "x is a prime number" as follows:

$$(1 < x) \wedge (\neg (\exists y \exists z ((y < x) \wedge (z < x) \wedge (x = (y * z))))) .$$

c) We can apply the so-called **priority rules**. For example, the priority rank of multiplications is supposed to be higher than the priority rank of additions. This rule allows writing x+y*z instead of the more formal x+(y*z) − because of its higher priority rank, multiplication must be "performed first". The most popular priority rules are the following:

c1) The priority rank of **function constants** is higher than the priority rank of **predicate constants**. This allows, for example, writing x*y = y*x instead of (x*y)=(y*x), or $x \in y \cup z$ − instead of $x \in (y \cup z)$ .

c2) The priority rank of **predicate constants** is higher than the priority rank of **propositional connectives and quantifiers**. This allows, for example, writing $y < x \wedge z < x$ instead of $(y < x) \wedge (z < x)$ .

c3) The priority rank of **quantifiers** is higher than the priority rank of **propositional connectives**. This allows, for example, writing $\exists x F \wedge \forall y G$ instead of $(\exists x(F)) \wedge (\forall y(G))$ , or writing $\neg \exists x F$ instead of $\neg (\exists x(F))$ .

c4) The priority rank of **negations** is higher than the priority rank of **conjunctions and disjunctions**. This allows, for example, writing $\neg A \wedge \neg B$ instead of $(\neg A) \wedge (\neg B)$ .

c5) The priority rank of **conjunctions and disjunctions** is higher than the priority rank of **implications**. This allows, for example, writing $A \to A \lor B$ instead of $A \to (A \lor B)$ .

In the usual mathematical practice some additional priority rules are used, but some of them are not allowed in the common programming languages. To avoid confusions do not use too many priority rules simultaneously!

According to the above priority rules, we can simplify the above expression "x is a prime number" obtaining a form that is much easier for human reading (but is somewhat complicated for computers to process it):

$$1 < x \land \neg \exists \, y \, \exists \, z \, (y < x \land z < x \land x = y * z) \quad .$$

As you see, all the above rules are mere abbreviations. In principle, you could use any other set of abbreviation rules accepted by your audience. If computers would do logic themselves, they would not need such rules at all (except, maybe, for displaying some of their results to humans, but why?).

**Exercise 1.2.6.** "Translate" the following assertions into our "language for people":

"x and y are siblings";
"x and y are brothers"; "x and y are sisters"; "x is cousin of y";
"parents of x and y are married";
"x and y are married people" (formalize both possible meanings);
"nobody is married to everybody".

But how about the predicate **Ancestor(x, y)** − "x is an ancestor of y"? Could it be expressed as a formula of our "language for people"? The first idea − let us "define" this predicate recursively:

$$\forall \, x \, \forall \, y \, (Father(x, y) \lor Mother(x, y) \to Ancestor(x, y)) \quad ;$$
$$\forall \, x \, \forall \, y \, \forall \, z \, (Ancestor(x, y) \land Ancestor(y, z) \to Ancestor(x, z)) \quad .$$

The second rule declares the *transitivity property* of the predicate. The above two formulas are **axioms**, allowing to derive the essential properties of the predicate *Ancestor(x, y)*. But how about a single formula F(x, y) in the "language for people", expressing that "x is an ancestor of y"? Such a formula should be a tricky combination of formulas *Father(x, y), Mother(x, y)* and *x=y.* And such a formula is **impossible**! For details see <u>Transitive predicates and recursion</u> below.

**Exercise 1.2.7.** "Translate" the following assertions into the language of first order arithmetic:

"x and y are <u>twin primes</u>" (examples of twin pairs: 3,5; 5,7; 11,13; 17,19;...),
"There are infinitely many pairs of twin primes" (the famous **Twin Prime Conjecture**),

"x is a power of 2" (Warning! ∃n(x=2$^n$) is not a correct solution. Why? Think about prime divisors of x instead.),
"Each positive even integer ≥4 can be expressed as a sum of two primes" (the famous <u>Goldbach Conjecture</u>).

**Free variables and bound variables**

Usually, people use these notions without definitions. Anyway, precise treatment follows.

The above expression "x is a prime number":

$$1 < x \land \neg \exists \, y \exists \, z (y < x \land z < x \land x = y * z)$$

contains 3 variables: x − occurs 4 times in terms, y − 2 times in terms and 1 time in quantifiers, z − occurs 2 times in terms and 1 time in quantifiers. Of course, x is here a "free" variable – in the sense that the "truth value" of the formula depends on particular "values" taken by x. On the contrary, the "truth value" of the formula does not depend on the particular "values" taken by the two "bound" variables y and z − the quantifiers ∃y, ∃z force these variables to "run across their entire range".

More precisely, first, we will count **only** the occurrences of variables in terms, not in quantifiers. And second, we will define a particular occurrence o$_x$ of a variable x in (a term of) a formula F as a **free occurrence** or a **bound occurrence** according to the following rules:

a) If F does not contain quantifiers ∃x, ∀x, then o$_x$ is free in F.

b) If F is ∃xG or ∀xG, then o$_x$ is bound in F.

c$_1$) If F is $G \land H, G \lor H$, or G→H, and o$_x$ is free in G (or in H), then o$_x$ is free in F.

c$_2$) If F is ¬G, ∃yG, or ∀yG, where y **is not** x, and o$_x$ is free in G, then o$_x$ is free in F.

d$_1$) If F is $G \land H, G \lor H$, or G→H, and o$_x$ is bound in G (or in H), then o$_x$ is bound in F.

d$_2$) If F is ¬G, ∃yG, or ∀yG (where y is any variable, x included), and o$_x$ is bound in G, then o$_x$ is bound in F.

Thus, the above formula $1 < x \land \neg \exists \, y \exists \, z (y < x \land z < x \land x = y * z)$ contains 4 free occurrences of x, 2 bound occurrences of y, and 2 bound occurrences of z.

**Exercise 1.2.8**. Verify that an occurrence of x in F cannot be free and bound

simultaneously. (Hint: assume that it is not the case, and consider the sequence of all sub-formulas of F containing this particular occurrence of x.)

Formally, we can use formulas containing free and bound occurrences of a single variable simultaneously, for example, $x > 1 \rightarrow \exists\ x(x > 1)$ . Or, many bound occurrences of a single variable, for example,

$$(\forall\ xF(x) \wedge \exists\ xG(x)) \vee \forall\ xH(x)$$

means the same as

$$(\forall\ xF(x) \wedge \exists\ yG(y)) \vee \forall\ zH(z)\ .$$

Still, we do not recommend using a single variable in many different roles in a single formula. Such formulas do not cause problems for computers, but they may become inconvenient for human reading.

Let us say, that x is a **free variable** of the formula F, if and only if F contains at least one free occurrence of x, or F does not contain occurrences of x at all.

If a formula contains free variables, i.e. variables that are not bound by quantifiers (for example: $x = 0 \vee x = 1$ ), then the "truth value" of such formulas may depend on particular values assigned to free variables. For example, the latter formula is "true" for x=1, yet it is "false" for x=2. Formulas that do not contain free occurrences of variables, are called **closed formulas**, for example:

$$\forall\ w \exists\ x(w < x \wedge prime(x))\ .$$

Closed formulas represent "definite assertions about objects of theory", they are expected to be (but not always really are) either "true", or "false".


**Term substitution**

To say that x is a free variable of the formula F, we may wish to write F(x) instead of simply F. Replacing all free occurrences of x by a term t yields an "instance" of the formula F. It would be natural to denote this "instance" by F(t).

For example, if F(x) is ∃y(y+y=x) and t is z*z+z, then F(t), or F(z*z+z) will denote ∃y(y+y=z*z+z).

However, if t would be y*y+y, then F(t), or F(y*y+y) would be ∃y(y+y=y*y+y). Is this really F(y*y+y)?

Thus, sometimes, substitutions can lead to crazy results. Another example: in our expression "x is a prime number", let us replace x by y. Will the resulting formula mean "y is a prime number"? Let's see:

$$1 < y \wedge \neg \exists\ y \exists\ z(y < y \wedge z < y \wedge y = y * z)\ .$$

Since y<y is always false, the second part ¬∃y∃z(...) is true, hence, the latter formula means simply that "1 is less than y", and not that "y is a prime number".

Of course, we failed because we replaced a free variable x by a variable y in such a way that some **free** occurrence of x became **bound** by a quantifiers for y (∃y). In this way we **deformed the initial meaning** of our formula.

The following simple rule allows to avoid such situations. Suppose, x is a free variable of the formula F. We will say that the **substitution F(x/t)** (i.e. the substitution of the term t for x in the formula F) **is admissible**, if and only if no free occurrences of x in F are located under quantifiers that bind variables contained in t. If the substitution F(x/t) is admissible, then, by replacing all free occurrences of x in F by t, of course, we do not change the initial meaning of the formula F(x), and hence, we may safely denote the result of this substitution by F(t).

**Exercise 1.2.9.** Is x/y an admissible substitution in the following formulas? Why?

$$x = 0 \lor \exists y (y > z) \quad ;$$
$$x = 0 \lor \exists y (y > x) \quad .$$

**Exercise 1.2.10** (optional). a) Mathematicians: think over the analogy between bound variables in logic and bound variables in sum expressions and integrals. b) Programmers: think over the analogy between bound variables in logic and loop counters in programs.

## 1.3. Axioms of Logic: Minimal System, Constructive System and Classical System

**The problem of reasoning**

Now we go on in detail to the second phase of formalization:

a) after having defined a *formal language* (predicate language) allowing to put down propositions about objects in our domain of interest,

b) and having formulated as *axioms* some of the propositions, that are "true" of the objects in the domain,

c) we must introduce some *means of reasoning* allowing to derive other statements that are as "true" of the objects as are the axioms.

Indeed, having formulated some fragment of our knowledge as a set of axioms $A_1$, ..., $A_n$ in some predicate language L, we do not think that $A_1$, ..., $A_n$ represent **all** statements that are "true" of the objects we are trying to

investigate. Many other statements **follow** from A$_1$, ..., A$_n$ as consequences.

**Example**. Assume, we have formulated the following axioms in our "language for people": $\neg \exists x (Male(x) \wedge Female(x))$ ; $\forall x (Male(x) \vee Female(x))$ , and the following facts: $Male(John); Female(Britney)$ . Then we do not need to formulate $\neg Female(John); \neg Male(Britney)$ as separate facts. These facts can be derived from the already registered facts by applying the axioms.

The problem of reasoning: "formula F follows from A$_1$, ..., A$_n$", what does it mean? The answer must be absolutely precise, since we wish to **teach reasoning to computers**.

**Solution of the problem**

First of all, let us notice that there are axioms and rules of inference that are applicable to any predicate languages, independently of the specific features of their domains. Such axioms and rules could be called "generally valid".

For example, assume, some formula F has the following form:

$$(B \rightarrow D) \rightarrow ((C \rightarrow D) \rightarrow (B \vee C \rightarrow D)) \quad ,$$

where B, C, D are some formulas. Then F is "true" independently of the specific facts represented in the formulas B, C, D.

Similarly, the following rule of inference MP (the famous *Modus Ponens*) is applicable independently of the facts represented in the formulas B, C:

Having derived the formulas B, B→C, derive the formula C.

If we have B→D and B→D already derived, then – by applying the rule MP to the above formula F – we can derive that BvC→D.

Now, we will try formulating a complete set of "generally valid" principles (axioms and rules of inference) of "logically correct reasoning". "Generally valid" means that these principles will be applicable in any predicate language. I.e. for any fixed predicate language L, we will formulate a uniform list of **logical axioms and inference rules** that would allow formalization of principles of reasoning that are "valid" for all languages. Such principles are called sometimes "pure logical" principles. Establishing of the existence of such general principles (and even, in a sense, a complete system of them) is the result of a 2500 year long history of great discoveries (or inventions?).

Aristotle (384-322 BC),

Gottlob Frege (1848-1925), Charles Sanders Peirce (1839-1914).

Bertrand Russell (1872-1970), David Hilbert (1862-1943).

**D.Hilbert, W.Ackermann**. Grundzüge der theoretischen Logik. Berlin (Springer), 1928 (see also: Hilbert and Ackermann's 1928 Logic Book by Stanley N. Burris).

The first version of logical axioms was introduced in 1879 by G. Frege in his above-mentioned *Begriffsschrift*. The next important version was proposed in 1910-1913 by B. Russell and A. Whitehead in their famous book *Principia Mathematica*. And finally, in 1928 D. Hilbert and W. Ackermann published in their above-mentioned book containing, in a sense, the final version of logical axioms. Equivalent modifications of this version are now used in all textbooks of mathematical logic.

In our version, the axioms will be represented by means of the so-called **axiom schemata** (programmers might call them *templates*). Each schema (template) represents an infinite, yet easily recognizable collection of single axioms. For example, schema $L_3$: $B \wedge C \rightarrow B$ may represent the following axioms ("instances of the schema") in the language of first order arithmetic:

$$x = y \wedge x = x \rightarrow x = y \quad,$$

$$1 * 1 = 1 \wedge 1 + 1 = 1 + 1 \rightarrow 1 * 1 = 1 \quad,$$

and many other axioms: take any formulas B, C in your predicate language, and you will obtain an instance of the axiom schema $B \wedge C \rightarrow B$ .

We will not specify properties of the **equivalence** connective in axioms. We will regard this connective as a derived one: $B \leftrightarrow C$ will be used as an abbreviation of $(B \rightarrow C) \wedge (C \rightarrow B)$ . Similarly, $A \, xor \, B$ can be used as an abbreviation of $\neg (B \leftrightarrow C)$ .

**Axioms of logic**

Suppose, we have specified some predicate language L. We adopt the following 15 axiom schemata as **the logical axioms for the language L.**

In the axiom schemata $L_1$-$L_{11}$ below, B, C and D are any formulas in the language L.

The first two axiom schemata $L_1$, $L_2$ represent the "definition" of the implication connective:

**L₁**: $B \rightarrow (C \rightarrow B)$ (what does it mean?),

**L₂**: $(B \rightarrow (C \rightarrow D)) \rightarrow ((B \rightarrow C) \rightarrow (B \rightarrow D))$ (what does it mean?).

Such (or similar) definition of implication is necessary, if we wish computers to handle implications correctly.

*Note*.The axioms $L_1$, $L_2$ represent the (currently) most popular version of "defining" the implication connective. About other (equivalent) versions − containing 3 or 4 axioms − see Hilbert, Bernays [1934] (Chapter III) and Exercise 1.5.2 below.

The following axiom schemata $L_3$–$L_5$ represent the "definition" of the AND-connective (conjunction):

**L₃**: $\quad B \wedge C \to B \quad$ (what does it mean?),

**L₄**: $\quad B \wedge C \to C \quad$ (what does it mean?),

**L₅**: $\quad B \to (C \to B \wedge C) \quad$ (what does it mean?).

Such (or similar) definition of conjunction is necessary, if we wish computers to handle conjunctions correctly.

The following axiom schemata $L_6$–$L_8$ represent the "definition" of the (**non-exclusive!**) OR-connective (disjunction):

**L₆**: $\quad B \to B \vee C \quad$ (what does it mean?),

**L₇**: $\quad C \to B \vee C \quad$ (what does it mean?),

**L₈**: $\quad (B \to D) \to ((C \to D) \to (B \vee C \to D)) \quad$ (what does it mean?).

Such (or similar) definition of disjunction is necessary, if we wish computers to handle disjunctions correctly.

**Warning!** Once again, as the above three axioms show, the disjunction connective is intended as the **non-exclusive "or"**: $B \vee C$ is meant as "B, or C, or both".

The next axiom schema $L_9$ represents the "definition" of the negation connective. In fact, it is a formal version of a proof method well-known in mathematics − refutation by deriving a contradiction (*Reductio ad absurdum*):

**L₉**: $\quad (B \to C) \to ((B \to \neg C) \to \neg B) \quad$ (what does it mean?).

*Note*. The axiom $L_9$ represents the (currently) most popular version of "defining" the negation connective. About other (equivalent) versions − see <u>Hilbert, Bernays [1934]</u> (Chapter III) and <u>Exercise 2.4.2</u> below.

The next axiom schema $L_{10}$ represents the famous principle "Contradiction Implies Anything" (*Ex contradictione sequitur quodlibet*, or *Ex falso sequitur quodlibet*):

**L₁₀**: $\quad \neg B \to (B \to C) \quad$ (what does it mean?).

The following axiom schema $L_{11}$ represents the famous **Law of Excluded Middle** (*Tertium non datur)*:

**L₁₁**: $\quad B \vee \neg B \quad$ (what does it mean?).

The above 11 schemata (plus the *Modus Ponens* rule of inference, see below) represent the **classical propositional logic** in the language L.

Now, the "definitions" of the universal and existential quantifiers follow.

In the following axiom schemata $L_{12}$, $L_{13}$, F is any formula, and t is a term such that F(x/t) is an admissible substitution (in particular, t may be x itself):

**L$_{12}$**: $\forall x F(x) \rightarrow F(t)$ (in particular, $\forall x F(x) \rightarrow F(x)$, what does it mean?),

**L$_{13}$**: $F(t) \rightarrow \exists x F(x)$ (in particular, $F(x) \rightarrow \exists x F(x)$, what does it mean?).

In the following schemata $L_{14}$, $L_{15}$, F is any formula, and G is a formula that does not contain x as a free variable:

**L$_{14}$**: $\forall x(G \rightarrow F(x)) \rightarrow (G \rightarrow \forall x F(x))$ (what does it mean?),

**L$_{15}$**: $\forall x(F(x) \rightarrow G) \rightarrow (\exists x F(x) \rightarrow G)$ (what does it mean?).

**Rules of inference**

In the following rules of inference, B, C and F are any formulas, and "... $\vdash$ ..." is read as "... prove(s) ...".

***Modus Ponens** (MP)*: B→C; B $\vdash$ C (what could this mean?).

***Generalization (Gen)***: F(x) $\vdash$ $\forall x F(x)$ (what could this mean?).

This list of logical axioms and rules of inference represents the so-called **classical predicate logic** in the predicate language L (or, simply − the **classical logic** in the language L).

---

**Possible misconceptions**

1. *Modus Ponens* rule allows to derive C, if we have B and B→C (C follows from B) already derived. The "converse" version B→C; C $\vdash$ B is wrong as a logical principle. Knowing B→C and C can serve as an argument in favor of B being true, but not as a reliable *proof* of B.

2. (Thanks to Ilmārs Cīrulis) No formal restrictions are put on the formula F(x) when applying *Generalization* rule. However, it will be useless to apply *Gen*, if you wish simply to explore the consequences of the formula F(x) as "a specific condition on the value of x". If one is applying *Gen* to F(x), one has in mind that F(x) is "true for all values of x".

---

**Some of the logical axioms are "wrong, but useful"!**

Three of the above axiom schemata seem to be (at least partly) problematic.

For example, how do you find the funny axiom $L_{10}$: $\neg B \rightarrow (B \rightarrow C)$ ? If $\neg B$ and B were true simultaneously, then anything were true? *Ex contradictione sequitur quodlibet?* Is this a really "true" axiom? Of course, it is not. Still, this does not matter: we do not need to know, were C "true" or not, if $\neg B$ and B were "true" simultaneously. By assuming that "if $\neg B$ and B were true simultaneously, then anything were true" **we greatly simplify our logical apparatus**. For example, we will prove in Section 2.6 that, in the classical logic, $\neg\neg B \rightarrow B$. This simple formula can't be proved without the "crazy" axiom $L_{10}$ (see Section 2.8).

In fact, the first axiom $L_1$: $B \rightarrow (C \rightarrow B)$ also is funny. If B is (unconditionally) true, then B follows from C, even if C has nothing in common with B? Moreover, in Exercise 1.4.2(d) we will see that the axioms $L_1$, $L_9$ allow proving that $\neg B$, B $\vdash \neg C$, i.e. if $\neg B$ and B were true simultaneously, then anything were false (thus, in a sense, $L_1$ contains already 50% of $L_{10}$!). After this, could we think of $L_1$ as a really "true" axiom? Of course, we can't. Still, this does not matter: if B is (unconditionally) true, then we do not need to know, follows B from C or not. By assuming that "if B is true, then B follows from anything" **we greatly simplify our logical apparatus**.

The above two phenomena are called **paradoxes of the material implication**, see Paradoxes of Material Implication by Peter Suber, and Falsity Implies Anything by Alexander Bogomolny.

May our decision to "greatly simplify" the logical apparatus have also some undesirable consequences? Let us consider the following formula F(x): $\forall y(child(x, y) \rightarrow Female(y))$. It seems, F(x) is intended to mean: "All the children of x are female". However, in our system of logic, F(x) is regarded as true also, if x does not have children at all! If you do not have children at all, then all your children are female! Or male? Or smart? Etc. Seems funny, but is, in fact, harmless...

Thus, it would be an exaggeration to call the above system of axioms and rules of inference **"a system of principles of correct reasoning"**. As we saw it, not all logical axioms represent such principles, some of them were introduced solely in order to simplify the system. It would be better to call this system **"an engine of good reasoning".** Though not perfect, it proved to be extremely efficient, much more efficient than the more complicated systems proposed in order to overcome its seeming deficiencies.

**Constructive logic**

Still, it appears that the most serious problem is caused not by $L_1$ and $L_{10}$: , it is caused by the axiom $L_{11}$: $B \vee \neg B$ − the Law of Excluded Middle. How can we think of $L_{11}$ as a "true" axiom, if (according to Gödel's Incompleteness Theorem) each sufficiently strong consistent theory contains undecidable propositions? We postulate that either B, or ¬B "must be true", yet for some B we will unable to prove neither B, nor ¬B! Knowing that $B \vee \neg B$ is "true" may inspire us to work on the problem, but it may appear useless, if we do not succeed... Should we retain $L_{11}$ as an axiom after this?

For this (and some other, see below) reasons some people reject $L_{11}$ as a "valid" logical axiom.

The above list of 15 axiom schemata as it stands is called the **classical logic**.

By excluding $L_{11}$ from the list the so-called **constructive** (historically, and in most textbooks − **intuitionistic**) **logic** is obtained. As a concept, it was introduced by <u>Luitzen Egbertus Jan Brouwer</u> (1881-1966) in 1908:

**L. E. J. Brouwer.** De onbetrouwbaarheid der logische principes (The unreliability of the logical principles), *Tijdschrift voor Wijsbegeerte*, 2 (1908), pp.152-158.

Brouwer's main objection was against non-constructive proofs which are enabled mainly by an "improper" use of the Law of Excluded Middle.

An elegant very short *non-constructive* proof proposes to use either $a = b = \sqrt{2}$ , or $a = \sqrt{2}; b = \sqrt{2}^{\sqrt{2}}$ to prove that there are two irrational numbers *a, b* such that $a^b$ is rational. The possibly shortest *constructive* proof of this fact is using $a = \sqrt{2}; b = 2\log_2 3$ instead.

**Exercise 1.3.1** (optional). Elaborate on these two proofs.

For the entire history, see <u>The Root-2 Proof as an Example of Non-constructivity</u> by **J. Roger Hindley**.

**Note.** A similar kind of non-constructive reasoning is represented by the so-called Double Negation Law: $\neg\neg B \rightarrow B$, see <u>Section 2.6</u>.

As a formal system, the intuitionistic logic was formulated by <u>Arend Heyting</u> in 1930:

**A. Heyting.** Die formalen Regeln der intuitionistischen Mathematik. *Sitzungsberichte der Preussischen Akademie der Wissenschaften, Physikalisch-mathematische Klasse*, 1930, pp.42-56.

The constructive concept of logic differs from the classical one mainly in its interpretation of disjunction and existence assertions:

− To prove BvC **constructively**, you must prove B, or prove C. To prove $B \vee C$ by using the classical logic, you are allowed to assume $\neg (B \vee C)$ as a hypothesis to derive a contradiction. Then, by the Law of Excluded Middle $(BvC) \vee \neg (BvC)$ you obtain $B \vee C$ . Having only such a "negative" proof, you may be unable to determine, which part of the disjunction BvC is true − B, or C, or both. Knowing that $B \vee C$ is "true" may inspire you to work on the problem, but it may appear useless, if you do not succeed…

− To prove $\exists x B(x)$ **constructively**, you must provide a particular value of x such that B(x) is true. To prove $\exists x B(x)$ by using the classical logic, you are allowed to assume $\forall x \neg B(x)$ as a hypothesis to derive a contradiction. Then, by the Law of Excluded Middle $\exists x B(x) \vee \neg \exists x B(x)$ you obtain $\exists x B(x)$ . Having only such a "negative" proof, you may be unable to find a particular x for which B(x) is true. Knowing that $\exists x B(x)$ is "true" may inspire you to work on the problem, but it may appear useless, if you do not succeed...

**Note**. As a joke, we may regard existence assertions as "huge disjunctions". For example, in the language of first order arithmetic, $\exists x B(x)$ could "thought" as $B(0) \vee B(1) \vee B(2) \vee ...$ , i.e. as an infinite "formula". Thus, the above two theses are, in a sense, "equivalent".

The constructive (intuitionist) logic is one of the great discoveries in mathematical logic − surprisingly, a complete system of constructive reasoning (as we will see later, in Section 4.4) can be obtained simply by dropping the Law of Excluded Middle from the list of valid logical principles.

See also Intuitionistic Logic by **Joan Moschovakis** in Stanford Encyclopedia of Philosophy.

Luitzen Egbertus Jan Brouwer: "He rejected in mathematical proofs the Principle of the Excluded Middle, which states that any mathematical statement is either true or false. In 1918 he published a set theory, in 1919 a measure theory and in 1923 a theory of functions all developed without using the Principle of the Excluded Middle." (according to MacTutor History of Mathematics archive).

See also Markov's Principle in Wikipedia.


**Minimal logic**

By excluding both $L_{10}$ and $L_{11}$ the so-called **minimal logic** is obtained. It was introduced by Ingebrigt Johansson in 1936:

**I.Johansson.** Der Minimalkalkül, ein reduzierter intuitionistischer Formalismus. *Compositio Mathematica*, 1936, Vol. 4, N1, pp.119-136.

As a separate concept, the minimal logic is much less significant than the constructive logic. Indeed, since it allows proving of ¬B, B ⊢ ¬C (in a sense,

50% of $L_{10}$!), dropping of $L_{10}$ is not a very big step.

**First order theories**

Having defined our predicate language L, and having formulated for L all the logical axioms and rules of inference, do we need more?

To complete the formalization of our informal vision of our domain of interest, we must formulate at least some specific axioms describing the *specific features of the domain*. Logical axioms and rules of inference are valid for any domains, i.e. they are "content-free" in the sense that, by using them only, one cannot derive specific information about the domain.

For example, one cannot derive from (any) logic, that

$$\forall\ x(Male(x)\vee Female(x))\quad.$$

To communicate this fact to the computer, we must formulate it as a *specific axiom*.

And, as we will prove in <u>Section 4.3</u>, **we will never need to introduce of specific rules of inference.** All that can be derived from the axioms of a first order theory, can be derived by using two logical rules of inference – *Modus Ponens* and *Generalization*.

Thus, as the result of the formalization process, we will obtain the so-called **first order theories** (or, knowledge bases, see below).

Each first order theory T includes:

a) a specific predicate **language** L(T);

b) **logical axioms** and rules of inference for this language (classical or constructive version may be adopted, see below);

c) a set of **specific (non-logical) axioms** of T (the specific knowledge, represented in T, if T is a knowledge base).

As the first example, let's use our "language for people" to build a "theory for people".

Examples of instances of logical axioms for the "language for people":

$L_1$:  $\forall\ x(Male(x)\rightarrow(Female(x)\rightarrow Male(x)))$  ;
$L_6$:  $\forall\ x\forall\ y(Mother(x,y)\rightarrow Mother(x,y)\vee Father(x,y))$  ;
$L_{11}$:  $Male(John)\vee\neg Male(John)$  ;
$L_{12}$:  $\forall x(Female(x))\rightarrow Female(Britney)$  ;
etc.

And let's introduce as many as possible non-logical axioms as well:

  $\forall\ x(Male(x)\vee Female(x))$  ;

$\neg\exists\, x(Male(x)\wedge Female(x))$   ;

$\forall\ x\forall\ y(Father(x,y)\rightarrow Male(x))$   ;

$\forall\ x(\forall\ y(\forall\ z((Father(x,z)\wedge Father(y,z))\rightarrow(x=y))))$   ...

**Exercise 1.3.2.** Extend this list of axioms as far as you can. Is your list complete? What do you mean by "complete"?

Another example of a first order theory — the so-called **first order arithmetic PA** (also called Peano arithmetic):

The language of PA:

a) The constants 0 and 1, and all variables are terms.

b) If $t_1$ and $t_2$ are terms, then $(t_1+t_2)$ and $(t_1*t_2)$ also are terms.

c) Atomic formulas are built as $(t_1=t_2)$, where $t_1$ and $t_2$ are terms.

Since we can use, for example, the expression $2x^2-3y^2-1=0$ as a shortcut for $(1+1)*x*x=(1+1+1)*y*y+1$, we can say simply that, in first order arithmetic, atomic formulas of are arbitrary Diophantine equations.

Instances of logical axioms for the language of first order arithmetic:

$L_1$:   $x=0\rightarrow(y=1\rightarrow x=0)$   ;

$L_6$:   $x=y\rightarrow x=y\vee z=1$   ;

$L_{11}$:   $0=1\vee\neg(0=1)$   ;

$L_{12}$:   $\forall x(x=1)\rightarrow x=1$   ;

etc.

The specific (non-logical) axioms of first order arithmetic:

x=x,
x=y→y=x,
x=y→(y=z→x=z),
x=y→x+1=y+1,
¬(0=x+1),
x+1=y+1→x=y,
x+0=x,
x+(y+1)=(x+y)+1,
x*0=0,
x*(y+1)=(x*y)+x,
   $B(0)\wedge\forall x(B(x)\rightarrow B(x+1))\rightarrow\forall xB(x)$   , where B is any formula in the language of arithmetic.

The axioms 7-10 represent recursive definitions of addition and multiplication. As the last the so-called **induction schema** is listed.

**Note**. But how about the exponentiation? Why didn't we include it in the language of arithmetic (plus the corresponding axioms: $x^0=1$; $x^{y+1}=(x^y)*x$)? This is not necessary, because it appears that exponentiation can be defined as a (very complicated) formula containing only addition and multiplication. See Mendelson [1997], or Podnieks [1997], Section 3.3.

For the most popular **axiom system of set theory** – see ZFC (Zermelo-Fraenkel's set theory).

**Proofs and theorems**

In general, any sequence of formulas $F_1$, $F_2$, ..., $F_m$ could be regarded as a (correct or incorrect) **formal proof** (or simply, a proof) of its last formula $F_m$. In a **correct proof**, formulas can play only the following roles:

a) *Axioms*. Some formulas may be instances of logical or non-logical axioms.

b) *Consequences* of earlier formulas, obtained by using rules of inference. For example, if $F_{25}$ is A, and $F_{34}$ is A→B, and $F_{51}$ is B, then we can say that $F_{51}$ has been obtained from $F_{25}$ and $F_{34}$ by using the *Modus Ponens* rule. Or, if $F_{62}$ is C(x), and $F_{63}$ is ∀xC(x), then we can say that $F_{63}$ has been obtained from $F_{62}$ by using the *Generalization* rule.

c) *Hypotheses*. Some formulas may appear in the proof without any formal justification, simply by assuming that they are "true".

Thus, the following notation can describe the actual status of a formal proof:

$$[T]: A_1, A_2, ..., A_n \vdash B,$$

where T is a first order theory (it determines which formulas are axioms and which are not), $A_1$, $A_2$, ..., $A_n$ are **all** the hypotheses used in the proof, and B is the formula proved by the proof. Each formula in such a proof must be either an axiom, or a hypothesis from the set $A_1$, $A_2$, ..., $A_n$, or it must be obtained from earlier formulas (in this proof) by using a rule of inference. You may read the above notation as "in theory T, by using formulas $A_1$, $A_2$, ..., $A_n$ as hypotheses, the formula B is proved".

As the first example, let us consider the following proof:

$$[L_5, MP]: \quad B, C \quad \vdash \quad B \wedge C \quad .$$

| (1) | $B$ | Hypothesis. |
|---|---|---|
| (2) | $C$ | Hypothesis. |
| (3) | $B \to (C \to B \wedge C)$ | It's the axiom schema $\mathbf{L_5}$. |
| (4) | $C \to B \wedge C$ | It follows from (1) and (3) by *Modus Ponens*. |
| (5) | $B \wedge C$ | It follows from (2) and (4) by *Modus Ponens*. |

For more serious examples of formal proofs see the next [Section 1.4](#) (Theorem 1.4.1 and Theorem 1.4.2).

In the real mathematical practice, when proving [T]: $A_1$, $A_2$, ..., $A_n$ ⊢ B, we may apply some theorem Q that already has been proved earlier. If we would simply insert Q into our formal proof, then, formally, this would yield only a proof of [T]: $A_1$, $A_2$, ..., $A_n$, Q ⊢ B, i.e. Q would be qualified as a hypothesis. To obtain the desired formal proof of [T]: $A_1$, $A_2$, ..., $A_n$ ⊢ B, we must insert not only Q itself, but **the entire proof** of Q! In this way we obtain the following

**Theorem 1.3.1.** If there is a proof [T]: $A_1$, $A_2$, ..., $A_n$, Q ⊢ B, and a proof [T]: $A_1$, $A_2$, ..., $A_n$ ⊢ Q, then there is a proof of [T]: $A_1$, $A_2$, ..., $A_n$ ⊢ B.

**Proof.** The proof of [T]: $A_1$, $A_2$, ..., $A_n$, Q ⊢ B is a sequence of formulas $F_1$, $F_2$, ..., Q, ..., $F_m$, B, and the proof of [T]: $A_1$, $A_2$, ..., $A_n$ ⊢ Q is some sequence of formulas $G_1$, $G_2$, ..., $G_p$, Q. Let us replace Q by $G_1$, $G_2$, ..., $G_p$, Q:

$$F_1, F_2, ..., G_1, G_2, ..., G_p, Q, ..., F_m, B,$$

and eliminate the duplicate formulas. This sequence is a proof of [T]: $A_1$, $A_2$, ..., $A_n$ ⊢ B. Q.E.D.

If, in some proof, hypotheses are not used at all, then we may write simply [T]: ⊢ B, or even T ⊢ B, and say that B is a **theorem** of theory T. Of course, by using axioms directly one almost never can prove really complicated theorems. Still, we can retain our simple formal definition of the notion of theorem because of the following

**Corollary 1.3.1.** If there is a proof of [T]: $A_1$, $A_2$, ..., $A_n$ ⊢ B, and proofs of [T]: ⊢ $A_1$, [T]: ⊢ $A_2$, ..., [T]: ⊢ $A_n$, then there is a proof of [T]: ⊢ B.

**Proof.** Immediately, by Theorem 1.3.1.


**Consistency**

Sometimes, a seemingly plausible set of axioms allows deriving of contradictions (the most striking example − [Russell's paradox](#) in the "naive" set theory). A formula F is called a **contradiction** in the theory T, if [T]: ⊢ F and [T]: ⊢ ¬F, i.e. if T proves and disproves F simultaneously. Theories allowing to derive contradictions are called **inconsistent theories**. Thus, T is called a **consistent theory**; if and only if T does not allow deriving of contradictions.

Normally, for a first order theory, the set of all theorems is infinite, and, therefore, **consistency cannot be verified empirically**. We may only hope to establish this desirable property by means of some **theoretical proof** (see Podnieks [1997], Section 5.4 for a more detailed discussion of this problem).

For theories adopting the above logical axioms, inconsistency is, in a sense, "the worst possible property". Indeed, the axiom $L_{10}$: $\neg B \rightarrow (B \rightarrow C)$ says that if a theory allows deriving a contradiction, then, in this theory, anything is provable. In Section 2.4 we will − without $L_{10}$ − prove 50% of it: $\neg B \rightarrow (B \rightarrow \neg C)$ . Thus, even without $L_{10}$ (but with $L_1$): if a theory allows deriving a contradiction, then, in this theory, anything is disprovable.

Is consistency enough for a theory to be "perfect", "non-empty" etc? In Section 4.3 we will prove the so-called Model Existence Theorem: if a first order theory is consistent, then there is a "model" (a kind of a "mathematical reality") where all its axioms and theorems are "true", i.e. a consistent theory is at least "non-empty".

## Completeness

If a formula contains free variables, i.e. variables that are not bound by quantifiers (for example: $x=0 \lor x=1$ ), then the "truth value" of such formulas may depend on particular values assigned to free variables. For example, the latter formula is "true" for x=1, yet it is "false" for x=2. Formulas that do not contain free occurrences of variables, are called **closed formulas**, for example:

$$\forall \, w \, \exists \, x (w < x \land prime(x)) \quad .$$

Closed formulas represent "definite assertions about the objects of a theory", and they are expected to be either "true", or "false". Or, in a first order **theory**, they are expected to be either provable, or disprovable (refutable). The above closed formula (stating that "there are infinitely many prime numbers") is provable − if our theory is first order arithmetic.

T is called a **complete theory**, if and only if for each closed formula F in the language of T: [T]: $\vdash$ F or [T]: $\vdash \neg$F, i.e. if and only if T proves or disproves any closed formula of its language. In other words: a complete theory can solve any problem from the domain of its competence.

In an **incomplete theory**, some closed formulas ("definite assertions about the objects of theory") can be neither proved, not disproved. Thus, an incomplete theory does not solve some of the problems from the domain of its competence.

Formally, according to this definition, an inconsistent theory is complete.

Indeed, the axiom $L_{10}$: $\neg B \rightarrow (B \rightarrow C)$ says that if a theory allows deriving a contradiction, then, in this theory, anything is provable, i.e. it is a complete theory.

Of course, if T would be both consistent and complete, then we could call it "absolutely perfect". Unfortunately, Gödel's incompleteness theorem says that all **fundamental mathematical theories are either inconsistent or incomplete**, i.e. none of them is "absolutely perfect" (see <u>Mendelson [1997]</u> or <u>Podnieks [1997]</u>, Section 6.1).

**Exercise 1.3.3** (optional). Re-formulate the above axiom system for a **many-sorted predicate language.**

## 1.4. The Flavor of Proving Directly

**Theorem 1.4.1**. $[L_1, L_2, MP]$: $\vdash A \rightarrow A$ for any formula A. What does it mean? It's the so-called **reflexivity property of implication**.

The following sequence of formulas represents a proof of the formula $A \rightarrow A$:

| | | |
|---|---|---|
| (1) | $(A\rightarrow((C\rightarrow A)\rightarrow A))\rightarrow((A\rightarrow(C\rightarrow A))\rightarrow(A\rightarrow A))$ | It's the axiom schema $L_2$: $(B\rightarrow(C\rightarrow D))\rightarrow((B\rightarrow C)\rightarrow(B\rightarrow D))$, with B = A, C = C$\rightarrow$A, D = A. |
| (2) | $A\rightarrow((C\rightarrow A)\rightarrow A)$ | It's the axiom schema $L_1$: $B\rightarrow(C\rightarrow B)$, with B = A, C = C$\rightarrow$A. |
| (3) | $(A\rightarrow(C\rightarrow A))\rightarrow(A\rightarrow A)$ | It follows from (1) and (2) by *Modus Ponens*. |
| (4) | $A\rightarrow(C\rightarrow A)$ | It's the axiom schema $L_1$: $B\rightarrow(C\rightarrow B)$, with B = A, C = C. |
| (5) | $A\rightarrow A$ | It follows from (3) and (4) by *Modus Ponens*. |

As you can see, the proof is easy to verify, but it could be hard to build it from scratch. "Why" should we take "the axiom $L_2$ with B = A, C = C$\rightarrow$A, D = A" for (1)?

**How could one invent a proof like the above one?** One of the versions could be as follows.

First, let's try to find an axiom, from which we could get A→A as a consequence. By trying $L_1$, i.e. B→(C→B), and setting B=C=A, we could obtain A→(A→A), a dead end, perhaps. So, let's try $L_2$, i.e. (B→(C→D))→((B→C)→(B→D)). By setting B=D=A we obtain (A→(C→A))→((A→C)→(A→A)). It seems to be a good decision − because the first premise A→(C→A) is, in fact, $L_1$. Hence, by applying the MP rule, we obtain (A→C)→(A→A). Now, how to make A→C "provable"? Since C is, in fact, an arbitrary formula, we can replace C by C→A, obtaining (A→(C→A))→(A→A). The premise is here, again, $L_1$, hence, by applying the MP rule, we obtain A→A. Q.E.D. By performing all our replacements at the very beginning, we obtain the above proof of the formula A→A. [BTW, the above two smart "operations" − obtaining A→A within $L_2$, and making $L_1$ of A→C, are applications of the so-called **unification**, a very general and very important method used in intellectual computer programs, for details, see .]

Hint. If, in your proof, you need to obtain an instance of the formula A→A, then just insert 5 formulas of the above proof (more precisely, insert the corresponding instances of these formulas).

**Theorem 1.4.2.** [$L_1$, $L_2$, MP]: A→B, B→C ⊢A→C, for any formulas A, B, C. What does it mean? It's the so-called **Law of Syllogism** (by Aristotle), or the **transitivity property of implication**.

The following sequence of formulas represents a proof of the formula A→C from the hypotheses A→B and B→C:

| | | |
|---|---|---|
| (1) | A→B | Hypothesis. |
| (2) | B→C | Hypothesis. |
| (3) | (A→(B→C))→((A→B)→(A→C)) | It's the axiom schema **$L_2$**: (B→(C→D))→((B→C)→(B→D)), with B = A, C = B, D = C. |
| (4) | (B→C)→(A→(B→C)) | It's the axiom schema **$L_1$**: B→(C→B), with B = B→C, C = A. |
| (5) | A→(B→C) | It follows from (2) and (4) by *Modus Ponens*. |
| (6) | (A→B)→(A→C) | It follows from (3) and (5) by *Modus Ponens*. |
| (7) | A→C | It follows from (1) and (6) by *Modus Ponens*. |

Hint. If, building a proof, you need to apply the transitivity property of

implication, then just insert the last 5 formulas of the above proof (more precisely, insert the corresponding instances of these formulas).

**Note.** Only axiom schemata $L_1$ and $L_2$ , and inference rule *Modus Ponens* are used for proving the Theorems 1.4.1 and 1.4.2. Hence, **these theorems will remain valid for any logical system containing $L_1$, $L_2$ and *Modus Ponens*.**

**Theorem 1.4.3.** a) $[L_1, L_2, MP]$: $A{\rightarrow}(B{\rightarrow}C) \vdash B{\rightarrow}(A{\rightarrow}C)$. What does it mean? It's the so-called **Premise Permutation Law**.

b) $[L_3, L_4, L_9, MP]$: $\vdash \neg(A \wedge \neg A)$ . What does it mean? It's the so-called **Law of Non-Contradiction**.

**Proof.** Do Exercise 1.4.1.

**Exercise 1.4.1.** Build sequences of formulas representing the following proofs (only the axiom schemata $L_1$ and $L_2$ and *Modus Ponens* are necessary):

a) $[L_1, MP]$: $A \vdash B{\rightarrow}A$ (a sequence of 3 formulas). What does it mean?

b) $[L_2, MP]$: $A{\rightarrow}B$, $A{\rightarrow}(B{\rightarrow}C) \vdash A{\rightarrow}C$ (a sequence of 5 formulas). What does it mean?

c) Prove Theorem 1.4.3(a). $[L_1, L_2, MP]$: $A{\rightarrow}(B{\rightarrow}C) \vdash B{\rightarrow}(A{\rightarrow}C)$ (a sequence of 9 formulas − thanks to Pavel Andreyev for the idea).

d) $[L_1, L_2, MP]$: $A{\rightarrow}(A{\rightarrow}B) \vdash A{\rightarrow}B$ (easy solution – a sequence of 9 formulas, a smarter solution proposed by Arnold Ostrovsky – 8 formulas). What does it mean?

e) (for smart students) $[L_1, L_2, MP]$: $\vdash (A{\rightarrow}(A{\rightarrow}B)){\rightarrow}(A{\rightarrow}B)$ (a sequence of 7 formulas! – a surprising sophisticated solution proposed by Ilmārs Cīrulis).

**Theorem 1.4.4.** $[L_{14}, MP, Gen]$ If F is any formula, and G is any formula that does not contain x as a free variable, then $G{\rightarrow}F(x) \vdash G{\rightarrow}\forall xF(x)$.

**Proof**. The following sequence of formulas represents a proof of the formula $G{\rightarrow} \forall xF(x)$ from the hypothesis G→F(x):

| | | |
|---|---|---|
| (1) | G→F(x) | Hypothesis. |
| (2) | ∀x(G→F(x)) | It follows from (1) by *Generalization*. |
| (3) | ∀x(G→F(x))→(G→∀xF(x)) | It's the axiom schema $\mathbf{L_{14}}$ (G does not contain x as a free variable). |

(4)  G→∀xF(x)

It follows from (2) and (3) by *Modus Ponens*.

**Exercise 1.4.2**. Build sequences of formulas representing the following proofs (F is any formula, and G is a formula that does not contain x as a free variable):

a) [$L_{15}$, MP, Gen]: F(x)→G ⊢ ∃xF(x)→G (a sequence of 4 formulas). What does it mean?

b) [$L_3$-$L_5$, MP]:  $A \wedge B$ ⊢ $B \wedge A$  (a sequence of 8 formulas). What does it mean?

c) [$L_6$-$L_8$, MP]: ⊢ $A \vee B \rightarrow B \vee A$  (a sequence of 5 formulas). What does it mean?

d) [$L_1$, $L_9$, MP]: B, ¬B ⊢ ¬C (a sequence of 9 formulas). What does it mean? It's a weak form of the "crazy" axiom **$L_{10}$**: ¬A→(A→B). This axiom says: "Contradiction implies anything". In the minimal logic we can prove 50% of $L_{10}$: "Contradiction implies that all is wrong". Of course, this 50%-provability of $L_{10}$ decreases the significance of the minimal logic accordingly.

e) Prove Theorem 1.4.3(b). [$L_3$, $L_4$, $L_9$, MP]: ⊢ $\neg(A \wedge \neg A)$  (a sequence of 5 formulas).

f) [$L_1$, $L_8$, $L_{10}$, MP]: ⊢ $\neg A \vee B \rightarrow (A \rightarrow B)$  (a sequence of 5 formulas). What does it mean?

g) [$L_8$, $L_{11}$, MP]: A→B, ¬A→B ⊢ B (a sequence of 7 formulas). What does it mean?

h) [$L_1$-$L_8$, MP]:  $A \rightarrow B$ ⊢ $A \vee C \rightarrow B \vee C$  (a sequence of 11 formulas). What does it mean?

i) [$L_1$-$L_{11}$, MP]: ⊢ $A \vee (A \rightarrow B)$  (a sequence of 14 formulas, a smarter solution by Ilmārs Cīrulis – 13 formulas). What does it mean? Does it mean anything at all?

**Exercise 1.4.3** (optional, for smart students). Could you build shorter sequences proving the formulas of Exercise 1.4.1(c, d) and Exercise 1.4.2(b), d)? Evgeny Vihrov verified in 2011 that any proof of the formula of Exercise 1.4.1(d) will take more than 5 formulas.

# 1.5. Deduction Theorems

If, by assuming B as a hypothesis, we have proved C, then we have proved that B implies C. This natural way of reasoning is formalized in the so-called **deduction theorems** (introduced by Jacques Herbrand and Alfred Tarski):

**J. Herbrand.** Recherches sur la théorie de la démonstration. PhD Thesis, University of Paris, 1930 (approved in April 1929).

**A. Tarski.** Ueber einige fundamentale Begriffe der Metamathematik. "Comptes Rendus de Séances de la Société des Sciences et des Lettres de Varsovie, Classe III", 1930, Vol.23, pp. 22-29.

We will prove two such theorems – Deduction Theorem 1 (for propositional logic) and Deduction Theorem 2 (for predicate logic).

**Theorem 1.5.1 (Deduction Theorem 1)**. If T is a first order theory, and there is a proof of

$$[T, MP]: A_1, A_2, ..., A_n, B \vdash C,$$

then there is a proof of

$$[L_1, L_2, T, MP]: A_1, A_2, ..., A_n \vdash B \rightarrow C.$$

I.e. having a *Modus Ponens* proof of C from the hypotheses $A_1, A_2, ..., A_n$, B, we can build a *Modus Ponens* proof of $B \rightarrow C$ from the hypotheses $A_1, A_2, ..., A_n$.

It appears that, usually, proving of $[T, MP]: ... B \vdash C$ is easier (technically simpler) than proving of $[T, MP]: ... \vdash B \rightarrow C$.

**Exercise 1.5.1** (optional, for smart students). Do not read the proof below. Try proving yourself.

**Proof** (thanks to Sergey Kozlovich for the idea, see also Kleene [1967], Exercise 10C). We must define a procedure allowing to convert any proof of $[T, MP]: A_1, A_2, ..., A_n, B \vdash C$ into a proof of $[L_1, L_2, T, MP]: A_1, A_2, ..., A_n \vdash B \rightarrow C$.

The easy way to do this would be using an induction by the number of formulas in the proof of $[T, MP]: A_1, A_2, ..., A_n, B \vdash C$. But we will use a more elegant idea. Any proof of $[T, MP]: A_1, A_2, ..., A_n, B \vdash C$ is a sequence of formulas $F_1, F_2, ...F_m$. We will replace each formula $F_i$ by 3 or 5 formulas, the last of these being the formula $B \rightarrow F_i$, retaining our sequence as a valid proof.

We must consider the following cases:

1) F is an axiom (i.e. an instance of a logical axiom or a non-logical axiom of

T). Replace F by 3 formulas: F, F→(B→F), B→F. The second formula is an instance of $L_1$, the third formula is obtained from the first two ones by using *Modus Ponens*.

2) F is one of the hypotheses $A_i$. Replace F by 3 formulas: F, F→(B→F), B→F. The second formula is an instance of $L_1$, the third formula is obtained from the first two ones by using *Modus Ponens*.

3) F is B. Replace F by the 5 formulas from the proof of Theorem 1.4.1, where D can be any formula:

(B→((D→B)→B))→((B→(D→B))→(B→B)) (an instance of $L_2$),

B→((D→B)→B) (an instance of $L_1$),

B→(D→B))→(B→B) (by *Modus Ponens*),

B→(D→B) (an instance of $L_1$),

B→B (by *Modus Ponens*).

The last formula is here, of course, B→F.

4) F is derived from some previous formulas $F_i$ and $F_j$ by *Modus Ponens*, $F_i$ having the form $F_j$→F (i.e. $F_j$→F and $F_j$ yield F by *Modus Ponens*). Then, the formulas
B→$F_j$,
B→($F_j$→F)
are already present in the converted proof (they appeared during the replacement operations applied to the formulas $F_j$ and $F_j$→F). So, replace F by 3 formulas:

(B→($F_j$→F))→((B→$F_j$)→(B→F)) (an instance of $L_2$),

(B→$F_j$)→(B→F) (by *Modus Ponens*),

B→F (by *Modus Ponens*).

Thus, what we have now, is a correct proof in $[L_1, L_2, MP]$ that is using the hypotheses $A_1, A_2, ..., A_n$, but not B! The last formula of this proof is B→C (because C is the last formula of our initial proof of $[L_1, L_2, MP]$: $A_1, A_2, ..., A_n, B \vdash C$). Thus, we have a proof of $[L_1, L_2, MP]$: $A_1, A_2, ..., A_n \vdash B→C$. Q.E.D.

The above proof of Deduction Theorem 1 includes, in fact, an **algorithm** allowing to obtain a proof of $[L_1, L_2, MP]$: $A_1, A_2, ..., A_n \vdash B→C$ from a given

proof of $[L_1, L_2, MP]$: $A_1, A_2, ..., A_n, B \vdash C$. The resulting proof is longer than the given one: if the given proof consists of *m* formulas, then the resulting proof consists of *3m* or *3m+2* formulas).

**Corollaries 1.5.1.** 1) If there is a proof of

$$[T, MP]: A_1, A_2, ..., A_n, B_1, B_2, ..., B_k \vdash C,$$

then there is a proof of

$$[L_1, L_2, T, MP]: A_1, A_2, ..., A_n \vdash (B_1 \rightarrow (B_2 \rightarrow (... \rightarrow (B_k \rightarrow C)...))).$$

2) If T includes (or proves) schemata $L_1$, $L_2$, then, if there is a proof of $[T, MP]$: $A_1, A_2, ..., A_n, B \vdash C$ then there is a proof of $[T, MP]$: $A_1, A_2, ..., A_n \vdash B \rightarrow C$.

In particular, if $[T, MP ]$: $B \vdash C$, then $[T, MP]$: $\vdash B \rightarrow C$.

And, if $[T, MP ]$: $B, C \vdash D$, then $[T, MP]$: $\vdash B \rightarrow (C \rightarrow D)$.

**Proof.** 1) By iterating Deduction Theorem 1.

2) If T is a theory which includes or proves the schemata $L_1$, $L_2$, then $[L_1, L_2, T, MP]$ is equivalent to $[T, MP]$. Q.E.D.

**Exercise 1.5.2** (optional, for smart students). In earlier versions of logical axioms, instead of the axiom $L_2$, in some texts, the following 3 axioms were in use:

$L_{21}$: $(A \rightarrow (A \rightarrow B)) \rightarrow (A \rightarrow B)$,

$L_{22}$: $(A \rightarrow (B \rightarrow C)) \rightarrow (B \rightarrow (A \rightarrow C))$ (i.e. the Premise Permutation Law),

$L_{23}$: $(A \rightarrow B) \rightarrow ((B \rightarrow C) \rightarrow (A \rightarrow C))$ (the Law of Syllogism, or the transitivity property of implication).

Verify that both versions, i.e. $[L_1, L_2, MP]$ and $[L_1, L_{21}, L_{23}, L_{23}, MP]$ are equivalent. (Hint: a) See Section 2.1 to verify that $[L_1, L_2, MP]$ proves $L_{21}$, $L_{23}$, $L_{23}$. b) Verify that $[L_1, L_{21}, L_{23}, L_{23}, MP]$ proves $L_2$ either directly, or by proving the Deduction Theorem 1 for $[L_1, L_{21}, L_{23}, L_{23}, MP]$.)

**Exercise 1.5.3** (optional, thanks to Sergey Kozlovich for the idea).

a) Prove the following "generalization" of the *Modus Ponens* rule:

$[L_1, L_2, MP]$: $(D_1 \rightarrow (D_2 \rightarrow ...(D_k \rightarrow B)...))$, $(D_1 \rightarrow (D_2 \rightarrow ...(D_k \rightarrow (B \rightarrow C))...)$ $\vdash$ $(D_1 \rightarrow (D_2 \rightarrow ...(D_k \rightarrow C)...))$.

b) Prove the following "generalization" of the axiom $L_{14}$ (formulas $D_1$, $D_2$, ..., $D_k$ do not contain x as a free variable):

[$L_1$, $L_2$, $L_{14}$, MP]: $\vdash$ $\forall x(D_1 \rightarrow (D_2 \rightarrow ...(D_k \rightarrow F(x))...)$ $\rightarrow$ $(D_1 \rightarrow (D_2 \rightarrow ... (D_k \rightarrow \forall x F(x))...)$.

**Exercise 1.5.4** (optional, for smart students). Investigate the size (the number of formulas) of the proof of [$L_1$, $L_2$, MP]: $A_1$, $A_2$, ..., $A_n$, $\vdash B \rightarrow C$ as a function f(m) of the size m of the proof of [$L_1$, $L_2$, MP]: $A_1$, $A_2$, ..., $A_n$, $B \vdash C$. You may wish to <u>report your result</u>. We will publish your report on the web as an appendix to this book. The current record holder is <u>Sergey Kozlovich, 2004</u>: $f(m) \leq 3m+2$. Improve this result, or prove that it is the best one possible.

**Exercise 1.5.5** (optional, for smart students). Investigate the size (the number of instances of **atomic** formulas) of the proof of [$L_1$, $L_2$, MP]: $A_1$, $A_2$, ..., $A_n$, $\vdash B \rightarrow C$ as a function g(m) of the size m of the proof of [$L_1$, $L_2$, MP]: $A_1$, $A_2$, ..., $A_n$, $B \vdash C$. You may wish to <u>report your result</u>. We will publish your report on the web as an appendix to this book. The current record holder is <u>Kirils Solovjovs, 2008</u>: $g(m, n) \leq 7m+24n-2$, where n is the number of instances of atomic formulas in the formula B. Improve this result, or prove that it is the best one possible.

**Warning!** *Generalization* **involved...**

Now, what, if in the proof of $A_1$, $A_2$, ..., $A_n$, $B \vdash C$ not only *Modus Ponens*, yet also *Generalization* is used?

We must be careful, because, trying "simply" to apply Deduction Theorem 1, we can obtain crazy results. Indeed, having a formula F(x), by *Generalization*, we obtain the formula $\forall x F(x)$. Thus, $F(x) \vdash \forall x F(x)$. If Deduction Theorem 1 could be extended to Gen without any restrictions, then we could conclude that $\vdash F(x) \rightarrow \forall x F(x)$. If this is true for any x, it is true also for x=2, hence, $\vdash F(2) \rightarrow \forall x F(x)$. Thus, if the number 2 is prime, then all numbers are prime?

So, let us try deriving a restricted formulation of the Deduction Theorem − it seems, we should **prohibit application of Gen to the free variables of B** − the hypothesis "to be moved".

**Theorem 1.5.2 (Deduction Theorem 2)**. If T is a first order theory, and there is a proof of

$$[\text{T, MP, Gen}]: A_1, A_2, ..., A_n, B \vdash C,$$

where *Generalization* is not applied to the free variables of B, then there is a proof of

$$[L_1, L_2, L_{14}, \text{T, MP, Gen}]: A_1, A_2, ..., A_n \vdash B \rightarrow C.$$

**Proof.** We must extend the above proof of the Deduction Theorem 1 that consisted of 4 cases. First, we must extend the first case:

1') F is an axiom (i.e. an instance of a logical axiom or a non-logical axiom of T). Replace F by 3 formulas: F, F→(B→F), B→F. The second formula is an instance of $L_1$, the third formula is obtained from the first two ones by using *Modus Ponens*.

And we must add the following case:

5) F is derived from some previous formula $F_i$ by *Generalization*, thus, F having the form $\forall x F_i$, where x is not free in the formula B. Replace F by the following 3 formulas:

$\forall x(B \rightarrow F_i) \rightarrow (B \rightarrow \forall x F_i)$,

$\forall x(B \rightarrow F_i)$,

$B \rightarrow \forall x F_i$.

Since x is not free in B, the first formula is an instance of $L_{14}$. The second formula is obtained by *Generalization* from the formula $B \rightarrow F_i$ that is already present in the converted proof (it appeared during the replacement operation applied to the formula $F_i$). The third formula is obtained from the first two ones by using *Modus Ponens*.

Thus, what we have now, is a correct proof in $[L_1, L_2, L_{14}, MP, Gen]$ that is using the hypotheses $A_1, A_2, ..., A_n$, but not B! The last formula of this proof is B→C (because C is the last formula our initial proof of $[L_1, L_2, L_{14}, MP, Gen]: A_1, A_2, ..., A_n, B \vdash C$). Thus, we have a proof of $[L_1, L_2, L_{14}, MP, Gen]: A_1, A_2, ..., A_n \vdash B \rightarrow C$. Q.E.D.

**Corollary 1.5.2**. 1) If there is a proof of

$$[T, MP, Gen]: A_1, A_2, ..., A_n, B_1, B_2, ..., B_k \vdash C,$$

where *Generalization* is not applied to the the free variables of the formulas $B_1, B_2, ..., B_k$, then there is a proof of

$$[L_1, L_2, L_{14}, T, MP, Gen]: A_1, A_2, ..., A_n \vdash (B_1 \rightarrow (B_2 \rightarrow (... \rightarrow (B_k \rightarrow C)...))).$$

2) If B is a **closed** formula, and there is a proof of

$$[T, MP, Gen]: A_1, A_2, ..., A_n, B \vdash C,$$

then there is a proof of

$$[L_1, L_2, L_{14}, T, MP, Gen]: A_1, A_2, ..., A_n \vdash B \rightarrow C.$$

3) If T is a theory whose axioms include schemata $L_1, L_2, L_{14}$, then, if there is

a proof of

$$[T, MP, Gen]: A_1, A_2, ..., A_n, B \vdash C,$$

where *Generalization* is not applied to the the free variables of B, then there is a proof of

$$[T, MP, Gen]: A_1, A_2, ..., A_n \vdash B \to C.$$

In particular, if [T, MP, Gen]: B ⊢ C, where *Generalization* is not applied to the free variables of B, then [T, MP, Gen]: ⊢ B→C.

**Proof.** Similar to the proof of the above Corollaries of Deduction Theorem 1.


**Warning! Previously proved theorems involved...**

In the real mathematical practice, when proving [T, MP, Gen]: $A_1, A_2, ..., A_n \vdash$ C, we may wish to apply some theorem Q that we have already proved earlier. If we would simply insert Q into our formal proof, then, formally, this would yield only that [T, MP, Gen]: $A_1, A_2, ..., A_n, Q \vdash$ C. To obtain the desired formal proof of [T, MP, Gen]: $A_1, A_2, ..., A_n \vdash$ C, we must insert not only Q itself, but **the entire proof** of Q!

Still, with the Deduction Theorem 2 this may be problematic. If we are proving [T, MP, Gen]: $A_1, A_2, ..., A_n, B \vdash$ C with the intention to apply Deduction Theorem 2 (to obtain [T, MP, Gen]: $A_1, A_2, ..., A_n \vdash$ B→C), then, before inserting the proof of Q, we must ensure that, in this proof, *Generalization* is not applied to the free variables of B. But, of course, the original proof of Q **could** contain such *Generalization*s! To solve this problem, we could try, in the proof of Q, before inserting it, rename simultaneously all the variables to which *Generalization* is applied and which are free variables in B. But this simultaneous renaming may affect the bound variables of Q, and thus − destroy the intended use of Q.

The problem is solved completely by the following extension of the Deduction Theorem 2:

**Theorem 1.5.3 (Deduction Theorem 2A).** If there is a proof of

$$[T, MP, Gen]: A_1, A_2, ..., A_n, B \vdash C,$$

where, **after B appears in the proof**, *Generalization* is not applied to the free variables of B, then there is a proof of

$$[L_1, L_2, L_{14}, T, MP, Gen]: A_1, A_2, ..., A_n \vdash B \to C.$$

Indeed, having such a theorem, we obtain the necessary

**Corollary 1.5.3.** If there is a proof of

$$[T, MP, Gen]: A_1, A_2, ..., A_n, B, Q \vdash C,$$

where, after B appears in the proof, *Generalization* is not applied to the free variables of B, and there is a proof of

$$[T, MP, Gen]: A_1, A_2, ..., A_n \vdash Q,$$

then there is a proof of

$$[T, MP, Gen]: A_1, A_2, ..., A_n \vdash B \rightarrow C.$$

**Proof of the Corollary.** In the proof of $[T, MP, Gen]: A_1, A_2, ..., A_n, B, Q \vdash C$, first, move all the hypotheses $A_1$, $A_2$, ..., $A_n$ to the beginning. Then, immediately after them, insert the proof of $[T, MP, Gen]: A_1, A_2, ..., A_n \vdash Q$. Now we have a proof of $[T, MP, Gen]: A_1, A_2, ..., A_n, B \vdash C$, where, after B appears in the proof, *Generalization* is not applied to the free variables of B. By Deduction Theorem 2A, then there is a proof of $[T, MP, Gen]: A_1, A_2, ..., A_n \vdash B \rightarrow C$. Q.E.D.

**Proof of the Deduction Theorem 2A.** Let us modify the above proof of the Deduction Theorem 2.

We must define a procedure allowing to convert any allowed proof of $[T, MP, Gen]: A_1, A_2, ..., A_n, B \vdash C$ into a proof of $[L_1, L_2, T, MP, Gen]: A_1, A_2, ..., A_n \vdash B \rightarrow C$.

Unlike the above proof, let us leave unchanged all the formulas of the proof of $[T, MP]: A_1, A_2, ..., A_n, B \vdash C$ **before B appears in the proof**. After this, **starting with B**, we will replace each formula F by 3 or 5 formulas, one of them being the formula $B \rightarrow F$.

We must consider the following cases:

1), 2), 3) − as in the proof of the Deduction Theorem 1.

4) F is derived from some previous formulas $F_i$ and $F_j$ by *Modus Ponens*, $F_i$ having the form $F_j \rightarrow F$ (i.e. $F_j \rightarrow F$ and $F_j$ yield F by *Modus Ponens*). Then, 4 subcases are possible.

4a) $F_j$ and $F_j \rightarrow F$ both appear before B, i.e. they remain unchanged in the converted proof. Let us replace F by the following 3 formulas: F, $F \rightarrow (B \rightarrow F)$, $B \rightarrow F$. The second formula is an instance of $L_1$, the third formula is obtained by using *Modus Ponens* from the first two ones.

4b) $F_j$ appears before B, and $F_j{\rightarrow}F$ is B or appears after B. Then, the formulas $F_j$ and $B{\rightarrow}(F_j{\rightarrow}F)$ are already present in the converted proof. Let us replace F by the following 5 formulas:

$(B{\rightarrow}(F_j{\rightarrow}F)){\rightarrow}((B{\rightarrow}F_j){\rightarrow}(B{\rightarrow}F))$ (an instance of $L_2$),

$(B{\rightarrow}F_j){\rightarrow}(B{\rightarrow}F)$ (by *Modus Ponens*),

$Fj{\rightarrow}(B{\rightarrow}Fj)$ (an instance of $L_1$),

$B{\rightarrow}Fj$ (by *Modus Ponens*),

$B{\rightarrow}F$ (by *Modus Ponens*).

4c) $F_j$ is B or appears after B, and $F_j{\rightarrow}F$ appears before B. Then, the formulas $B{\rightarrow}F_j$ and $F_j{\rightarrow}F$ are already present in the converted proof. Let us replace F by the following 5 formulas from the proof of Theorem 1.4.2:

$(B{\rightarrow}(F_j{\rightarrow}F)){\rightarrow}((B{\rightarrow}F_j){\rightarrow}(B{\rightarrow}F))$ (an instance of $L_2$),

$(F_j{\rightarrow}F){\rightarrow}(B{\rightarrow}(F_j{\rightarrow}F))$ (an instance of $L_1$),

$B{\rightarrow}(F_j{\rightarrow}F)$ (by *Modus Ponens*),

$(B{\rightarrow}F_j){\rightarrow}(B{\rightarrow}F)$ (by *Modus Ponens*),

$B{\rightarrow}F$ (by *Modus Ponens*).

4d) $F_j$ and $F_j{\rightarrow}F$ both are B or appear after B. Then, the formulas $B{\rightarrow}F_j$ and $B{\rightarrow}(F_j{\rightarrow}F)$ are already present in the converted proof (they appeared during the replacement operations applied to the formulas $F_j$ and $F_j{\rightarrow}F$). Let us replace F by the following 3 formulas:

$(B{\rightarrow}(F_j{\rightarrow}F)){\rightarrow}((B{\rightarrow}F_j){\rightarrow}(B{\rightarrow}F))$ (an instance of $L_2$),

$(B{\rightarrow}F_j){\rightarrow}(B{\rightarrow}F)$ (by *Modus Ponens*),

$B{\rightarrow}F$ (by *Modus Ponens*).

5) F is derived from some previous formula $F_i$ by *Generalization*, thus, F is in the form $\forall xF_i$. Then, 2 subcases are possible.

5a) $F_i$ appears before B. Then x is not free in B. Let us replace F by the following 3 formulas:

F (by *Generalization*, x is not free in B),

$F{\rightarrow}(B{\rightarrow}F)$ (an instance of $L_1$),

B→F

5b) $F_i$ is B or appears after B. Then x is not free in B, and the formula $B{\rightarrow}F_i$ that is already present in the converted proof (it appeared during the replacement operation applied to the formula $F_i$). Let us replace F by the following 3 formulas:

$\forall x(B{\rightarrow}F_i)$ (by *Generalization*, x is not free in B),

$\forall x(B{\rightarrow}F_i){\rightarrow}(B{\rightarrow}\forall xF_i)$ (an instance of $L_{14}$, since x is not free in B),

$B{\rightarrow}\forall xF_i$ (by *Modus Ponens*).

Thus, what we have now, is a correct proof in $[L_1, L_2, L_{14}, T, MP, Gen]$ that is using the hypotheses $A_1, A_2, ..., A_n$, but not B! The last formula of this proof is B→C (because C is the last formula our initial proof of [T, MP, Gen]: $A_1$, $A_2, ..., A_n, B \vdash C$). Thus, we have a proof of $[L_1, L_2, L_{14}, T, MP, Gen]$: $A_1, A_2$, ..., $A_n \vdash B{\rightarrow}C$. Q.E.D.

**Exercise 1.5.6** (optional, for smart students). In some other textbooks, a somewhat different system of logical axioms is used: instead of the axioms $L_{14}$, $L_{15}$ and the *Generalization* rule the following two rules of inference are used:

$G{\rightarrow}F(x) \vdash G{\rightarrow}\forall xF(x)$ ($\forall$-Introduction);

$F(x){\rightarrow}G \vdash \exists xF(x){\rightarrow}G$ ($\exists$-Elimination).

Of course, here, G is a formula that does not contain x as a free variable. Verify that both systems are equivalent in all of their versions (minimal, constructive, and classical).

# 2. Propositional Logic

George Boole (1815-1864): "In 1854 he published *An Investigation into the Laws of Thought, on Which are founded the Mathematical Theories of Logic and Probabilities.* Boole approached logic in a new way reducing it to a simple algebra, incorporating logic into mathematics. He pointed out the analogy between algebraic symbols and those that represent logical forms. It began the algebra of logic called Boolean algebra which now finds application in computer construction, switching circuits etc." (according to MacTutor History of Mathematics archive).

See also:

**G.Boole**. The Calculus of Logic. *The Cambridge and Dublin Mathematical Journal*, vol. 3 (1848) (available online, published by **David R. Wilkins**).

**The surprising efficiency of Deduction Theorem 1**

Try proving of

$$[L_1 - L_4, MP]: \vdash (A \to (B \to C)) \to (A \land B \to C) . \qquad (0)$$

For proving directly – almost an impossible task!

But now, having Deduction Theorem 1, we can simplify the task of proving (0) and make it feasible. More precisely – the task **of proving that (0) is provable.** We will not propose a full proof of (0), we will propose a shorter proof that such a full proof exists!

Indeed, by introducing two hypotheses, we can proceed as follows:

| | | |
|---|---|---|
| (1) | $A \to (B \to C)$ | Hypothesis. |
| (2) | $A \land B$ | Hypothesis. |
| (3) | $A \land B \to A$ | Axiom **L$_3$**: $B \land C \to B$ with B = A, C = B. |
| (4) | $A \land B \to B$ | Axiom **L$_4$**: $B \land C \to C$ with B = A, C = B. |
| (5) | A | By MP, from (2), (3). |
| (6) | B | By MP, from (2), (4). |
| (7) | B→C | By MP, from (1), (5). |
| (8) | C | By MP, from (6), (7). |

Thus, we have established that

$$[L_3\text{-}L_4, MP]: \quad A \to (B \to C), A \land B \quad \vdash \quad C$$

Now, by Deduction Theorem 1,

$$[L_1\text{-}L_4, MP]: \vdash \quad A \to (B \to C) \quad \vdash \quad A \land B \to C$$

And let us apply this theorem once more,

$$[L_1\text{-}L_4, MP]: \vdash \quad (A \to (B \to C)) \to (A \land B \to C) \quad .$$

**Note.** In fact, we proved here only $A \to (B \to C), A \land B \vdash C$, but **we did not prove (0), i.e. we did not produce a sequence of formulas proving (0). We just proved that such a sequence does exist!** To produce it really, we must apply (twice!) the algorithm described in the proof of Deduction Theorem 1. As the result, we will obtain a proof consisting of $3(3*8+2)+2=80$ formulas!

## 2.1. Proving Formulas Containing Implication only

**Exercise 2.1.1.** a) Produce a sequence of 4 formulas proving

$$[L_1, L_2, MP]: A \to (A \to B), A \vdash B.$$

b) Apply the algorithm described in the proof of Deduction Theorem 1 producing a sequence of $3*4+2=14$ formulas proving

$$[L_1, L_2, MP]: A \to (A \to B) \vdash A \to B.$$

c) Apply the algorithm once more – producing a sequence of 44 formulas proving

$$[L_1, L_2, MP]: \vdash (A \to (A \to B)) \to (A \to B).$$

Compare these numbers with the currently best solution proposed by human solvers – a sophisticated sequence of 7 formulas instead of 44 ones! See Exercise 1.4.1(e).

**Warning!** Always be careful when selecting hypotheses. For example, to prove the strange formula (the so-called Peirce's Law) $\vdash ((A \to B) \to A) \to A$ (it **is** provable in the classical logic, **not** in the constructive logic!), you can try proving that $(A \to B) \to A \vdash A$, but not $A \to B, A \vdash A$. Why? Because, by Deduction Theorem 1, from $A \to B, A \vdash A$ it follows that $A \to B \vdash A \to A$ and $\vdash (A \to B) \to (A \to A)$, or $A \vdash (A \to B) \to A$ and $\vdash A \to ((A \to B) \to A)$. Where do you see $\vdash ((A \to B) \to A) \to A$ here?

**Exercise 2.1.2.** Prove the following [$L_1$, $L_2$, MP]:

a) $\vdash$((A→B)→(A→C))→(A→(B→C)). What does this formula mean?

b) $\vdash$(A→B)→((B→C)→(A→C)). What does this formula mean? It's another version of the so-called **Law of Syllogism** (by Aristotle), or the **transitivity property of implication**. Explain the difference between this formula and Theorem 1.4.2: A→B, B→C $\vdash$A→C.

c) $\vdash$(A→(B→C))→(B→(A→C)). What does this formula mean? It's another version of the so-called **Premise Permutation Law.** Explain the difference between this formula and Exercise 1.4.1(c): A→(B→C) $\vdash$B→(A→C).

## 2.2. Proving Formulas Containing Conjunction

The following simple theorem allows to make some proofs shorter:

**Theorem 2.2.1**. a) [$L_5$, MP] A, B $\vdash$ $A \wedge B$ .

b) [$L_3$, $L_4$, MP]: $A \wedge B$ $\vdash$A, $A \wedge B$ $\vdash$B.

c) If [T, MP]: A, B $\vdash$ C, then [T, $L_3$, $L_4$, MP]: $A \wedge B$ $\vdash$C.

Let us prove (a).

| | | |
|---|---|---|
| (1) | A | Hypothesis. |
| (2) | B | Hypothesis. |
| (3) | $A \rightarrow (B \rightarrow A \wedge B)$ | Axiom **L$_5$**: $B \rightarrow (C \rightarrow B \wedge C)$ with B = A, C = B. |
| (4) | $B \rightarrow A \wedge B$ | By MP, from (1) and (3). |
| (5) | $A \wedge B$ | By MP, from (2) and (4). |

**Exercise 2.2.1.** Prove (b, c) of Theorem 2.2.1.

Theorem 2.2.1 allows for easy proving of **equivalences**. Let us remind that B↔C is defined as a shortcut for $(B \rightarrow C) \wedge (C \rightarrow B)$ . Of course, we will call B and C equivalent formulas, if and only if $\vdash$B↔C. For example, by Theorem 1.4.1, [$L_1$, $L_2$, MP] $\vdash$A→A, hence, [$L_1$, $L_2$, $L_5$, MP] $\vdash$ $(A \rightarrow A) \wedge (A \rightarrow A)$ , i.e.

$$[L_1, L_2, L_5, MP] \vdash A \leftrightarrow A.$$

**Theorem 2.2.2**. a) [$L_1$, $L_2$, $L_5$, MP] $\vdash$ (A→(B→C)) ↔ ((A→B)→(A→C)) (extension of the axiom $L_2$).

b) [$L_1$, $L_2$, $L_5$, MP] $\vdash$ (A→(B→C))↔(B→(A→C)) (extension of the **Premise Permutation Law**).

c) [$L_1$-$L_4$, MP] $\vdash$ $(A \rightarrow B) \wedge (B \rightarrow C) \rightarrow (A \rightarrow C)$ ( another form of the **Law of Syllogism**, or **transitivity property of implication**).

**Proof**. a) Of course, (a) of the Exercise 2.1.2 is the reverse formula of the axiom $L_2$. Hence, by Theorem 2.2.1 we obtain (a).

b) By (c) of the Exercise 2.1.2, and Theorem 2.2.1.

c)

| (1) | $(A \rightarrow B) \wedge (B \rightarrow C)$ | Hypothesis. |
|---|---|---|
| (2) | $(A \rightarrow B) \wedge (B \rightarrow C) \rightarrow (A \rightarrow B)$ | Axiom **L$_3$**: $B \wedge C \rightarrow B$ with B = A→B, C = B→C. |
| (3) | $(A \rightarrow B) \wedge (B \rightarrow C) \rightarrow (B \rightarrow C)$ | Axiom **L$_4$**: $B \wedge C \rightarrow C$ with B = A→B, C = B→C. |
| (4) | A→B | By MP, from (1) and (2). |
| (5) | B→C | By MP, from (1) and (3). |
| (6) | A→C | By by the transitivity property of implication (Theorem 1.4.2). |

Thus, we have established that [$L_1$-$L_4$, MP]: $(A \rightarrow B) \wedge (B \rightarrow C)$ $\vdash$ A→C. By Deduction Theorem 1, [$L_1$-$L_4$, MP] $\vdash$ $((A \rightarrow B) \wedge (B \rightarrow C)) \rightarrow (A \rightarrow C)$ .

**Exercise 2.2.2.** Prove the following [$L_1$- $L_5$, MP]:

a) A→B, A→C $\vdash$ $A \rightarrow B \wedge C$ . What does it mean?

b) $\vdash$ $(A \rightarrow B) \wedge (A \rightarrow C) \rightarrow (A \rightarrow B \wedge C)$ . What does it mean?

c) $A \rightarrow B \wedge C$ $\vdash$ A→B. What does it mean?

d) $A \rightarrow B \wedge C$ $\vdash$ A→C. What does it mean?

e) $\vdash$ $(A \rightarrow B \wedge C) \rightarrow (A \rightarrow B) \wedge (A \rightarrow C)$ .(Hint: apply transitivity of implication.) What does it mean? Hence,

$$[L_1\text{-} L_5, \text{MP}]: \vdash (A \rightarrow B \wedge C) \leftrightarrow (A \rightarrow B) \wedge (A \rightarrow C) \quad .$$

**Theorem 2.2.3** (properties of the conjunction connective). [L$_1$- L$_5$, MP]:

a) ⊢ $A \land B \leftrightarrow B \land A$ . What does it mean? **Conjunction is commutative**.

b) ⊢ $A \land (B \land C) \leftrightarrow (A \land B) \land C$ . What does it mean? **Conjunction is associative**.

c) ⊢ $A \land A \leftrightarrow A$ . What does it mean? **Conjunction is idempotent.**

**Proof.** Do Exercise 2.2.3.

Let us remind that the equivalence connective A↔B is defined as a shortcut for $(A \rightarrow B) \land (B \rightarrow A)$ .

**Theorem 2.2.4** (properties of the equivalence connective). [L$_1$- L$_5$, MP]:

a) ⊢A↔A (**reflexivity**),

b) ⊢(A↔B)→(B↔A) (**symmetry**),

c) ⊢(A↔B)→((B↔C) →((A↔C)) (**transitivity**).

**Proof.** Do Exercise 2.2.5.

**Exercise 2.2.3.** Prove Theorem 2.2.3.

**Exercise 2.2.4.** Prove the following, [L$_1$- L$_5$, MP]:

a) ⊢ $(A \rightarrow (B \rightarrow C)) \leftrightarrow (A \land B \rightarrow C)$ . What does it mean?

b) ⊢ $(A \rightarrow B) \rightarrow (A \land C \rightarrow B \land C)$ . What does it mean? The converse formula $(A \land C \rightarrow B \land C) \rightarrow (A \rightarrow B)$ cannot be true. Explain, why.

c) A ⊢ $B \leftrightarrow B \land A$ . What does it mean?

**Exercise 2.2.5**. Prove Theorem 2.2.4.


## 2.3. Proving Formulas Containing Disjunction


The following theorems allows to make some proofs greatly shorter.

**Theorem 2.3.0.** [L$_6$, MP]: A ⊢ $A \lor B$ ; [L$_7$, MP]: B ⊢ $A \lor B$ .

**Proof**. Do Exercise 2.3.1.

**Theorem 2.3.1.** [L$_1$, L$_2$, L$_8$, MP]: If there is a proof of

$$A_1, A_2, ..., A_n, B \vdash D,$$

and a proof of

$$A_1, A_2, ..., A_n, C \vdash D,$$

then there is a proof of

$$A_1, A_2, ..., A_n, B \vee C \vdash D.$$

**Proof**. By Deduction Theorem 1, there are proofs of $A_1$, $A_2$, ..., $A_n \vdash B{\to}D$ and $A_1$, $A_2$, ..., $A_n \vdash C{\to}D$. Write down these two proofs one after another, append the axiom $L_8$ and $B \vee C$ as a hypothesis, and apply (three times) MP. Q.E.D.

**Theorem 2.3.2**. a) [ $L_5$, $L_6$-$L_8$, MP]: $\vdash A \vee B \leftrightarrow B \vee A$ . What does it mean? **Disjunction is commutative**.

b) [$L_1$, $L_2$, $L_5$, $L_6$-$L_8$, MP]: $\vdash A \vee A \leftrightarrow A$ . What does it mean? **Disjunction is idempotent**.

**Proof.** Do Exercise 2.3.1.

**Theorem 2.3.3. Disjunction is associative:**

$$[L_1, L_2, L_5, L_6\text{-}L_8, \text{MP}]: \vdash A \vee (B \vee C) \leftrightarrow (A \vee B) \vee C .$$

**Proof**. Indeed, to prove, for example,

$$A \vee (B \vee C) \vdash (A \vee B) \vee C ,$$

we can first prove $A \vdash (A \vee B) \vee C$ and $B \vee C \vdash (A \vee B) \vee C$ and after that – apply Theorem 2.3.1. The second part of the proof can be further reduced in the same way. Continue the proof in Exercise 2.3.2.

**Exercise 2.3.1.** Prove the following:

a) Theorem 2.3.0.

b) [$L_8$, MP]: A→C, B→C $\vdash A \vee B \to C$ . What does it mean?

c) Theorem 2.3.2.

**Exercise 2.3.2.** a) Complete the proof of Theorem 2.3.3($\to$).

b) Prove ($\leftarrow$) of Theorem 2.3.3.

**Exercise 2.3.3.** a) Prove that

[$L_1$, $L_2$, $L_6$-$L_8$, MP]: $\vdash (A \to B) \to (A \vee C \to B \vee C)$ . What does it mean? The converse formula $(A \vee C \to B \vee C) \to (A \to B)$ cannot be true. Explain, why.

c) Prove that

[$L_1$, $L_2$, $L_6$-$L_8$, MP]: $\vdash$A→B, C→D $\vdash A \vee C \to B \vee D$ . What does it mean?

The following theorem corresponds to the well-known **distributive property** of (number) addition to multiplication: (a+b)c = ac+bc. Of course, the "dual" distributive property (of multiplication to addition) does not hold for numbers: ab+c=(a+c)(b+c) would imply ab+c=ab+ac+bc+cc, c=ac+bc+cc, and, if c<>0, then 1=a+b+c. Still, surprisingly, in logic,

**Theorem 2.3.4. Conjunction is distributive to disjunction, and disjunction is distributive to conjunction:**

$$\text{a) } [L_1\text{-}L_8, MP]: \vdash (A \wedge B) \vee C \leftrightarrow (A \vee C) \wedge (B \vee C) \ .$$
$$\text{b) } [L_1\text{-}L_8, MP]: \vdash (A \vee B) \wedge C \leftrightarrow (A \wedge C) \vee (B \wedge C) \ .$$

**Exercise 2.3.4.** (Hint: apply Theorem 2.3.1).

a) Prove (a→) of Theorem 2.3.4.

b) Prove (a←) of Theorem 2.3.4 (Hint: apply twice Theorem 2.3.1).

c) Prove (b→) of Theorem 2.3.4.

d) Prove (b←) of Theorem 2.3.4.


# 2.4. Formulas Containing Negation – Minimal Logic


The following theorem allows to make some proofs greatly shorter:

**Theorem 2.4.1**. Having a proof of

$$[L_1, L_2, L_9, MP]: A_1, A_2, ..., A_n, B \vdash C,$$

and a proof of

$$[L_1, L_2, L_9, MP]: A_1, A_2, ..., A_n, B \vdash \neg C,$$

we can build a proof of

$$[L_1, L_2, L_9, MP]: A_1, A_2, ..., A_n \vdash \neg B.$$

What does this mean?

**Proof.** a) By Deduction Theorem 1, there are proofs of $A_1, A_2, ..., A_n \vdash B \rightarrow C$ and $A_1, A_2, ..., A_n \vdash B \rightarrow \neg C$. Write down these two proofs one after another, append the axiom $L_9$, and apply (twice) MP. Q.E.D.

**Exercise 2.4.1.** Prove the following:

a) $[L_1, L_2, L_9, MP]$: A, ¬B ⊢ ¬(A→B). Or, $[L_1\text{-}L_4, L_9, MP]$: ⊢ $A \wedge \neg B \rightarrow \neg (A \rightarrow B)$ . What does it mean?

b) ⊢[L$_1$, L$_2$, L$_9$, MP]: (A→¬A)→¬A. What does it mean?

**Exercise 2.4.1A** (optional, for smart students) Investigate the size (the number of formulas) of the proof of [L$_1$, L$_2$, L$_9$, MP]: A$_1$, A$_2$, ..., A$_n$, ⊢¬B as a function f(k, m) of the sizes k, m of the proofs of [L$_1$, L$_2$, L$_9$, MP]: A$_1$, A$_2$, ..., A$_n$, B ⊢C and $_1$, A$_2$, ..., A$_n$, B ⊢¬C. You may wish to report your result. We will publish your report on the web as an appendix to this book. The current record holder is Aiga Romane, 2008: f(k, m) ≤ 3(k+m)+7. Improve this result, or prove that it is the best possible one.

**Attention: non-constructive reasoning!** In Section 2.6, we will use the classical logic [L$_1$-L$_{11}$, MP] to prove the converse formula of (b): $\neg(A \rightarrow B) \rightarrow A \wedge \neg B$ , i.e. the equivalence $\neg(A \rightarrow B) \leftrightarrow A \wedge \neg B$ . This formula cannot be proved in the constructive logic [L$_1$-L$_{10}$, MP] (see Section 2.8).

**Theorem 2.4.2.** [L$_1$, L$_2$, L$_9$, MP]: ⊢(A→B)→(¬B→¬A). What does it mean? It's the so-called **Contraposition Law**.

**Proof.** Do Exercise 2.4.2.

**Note**. The following form of Theorem 2.4.2 is called **Modus Tollens**:

$$[L_1, L_2, L_9, MP]: \vdash A \rightarrow B, \neg B \vdash \neg A.$$

**Attention: non-constructive reasoning!** In Section 2.6, we will use the classical logic [L$_1$-L$_{11}$, MP] to prove the converse formula (¬B→¬A)→(A→B), i.e. the equivalence (A→B)↔(¬B→¬A). We will see also that these formulas cannot be proved in the constructive logic [L$_1$-L$_{10}$, MP] (see Section 2.8).

**Exercise 2.4.2.** a) Prove Theorem 2.4.2 in two ways: by using Theorem 2.4.1, and without it.

b) (optional, for smart students) Verify that, in our axiom system, the Law of Non-Contradiction and the Contraposition Law could be used instead of the axiom L$_9$. More precisely: prove L$_9$ in the logic [L$_1$-L$_5$, Law of Non-Contradiction, Contraposition Law, MP]. Be careful: do not use theorems depending on the axiom L$_9$.

**Theorem 2.4.3.** [L$_1$-L$_9$, MP]: ⊢(A→¬B)↔(B→¬A). What does it mean?

**Attention: non-constructive reasoning!** The formula (¬A→B)↔(¬B→A) (very similar to the formula of Theorem 2.4.3) can be proved only in the classical logic! See Section 2.8.

**Proof.** Do Exercise 2.4.3.

**Theorem 2.4.4.** [L$_1$, L$_2$, L$_9$, MP]: ⊢A→¬¬A. What does it mean?

**Attention: non-constructive reasoning!** In Section 2.6, we will use the

classical logic [$L_1$-$L_{11}$, MP] to prove the converse formula ⊢¬¬A→A, i.e. the equivalence ⊢ ¬¬A↔A (the so-called **Double Negation Law**). We will see also (Section 2.8) that the formula ¬¬A→A cannot be proved in the constructive logic [$L_1$-$L_{10}$, MP].

**Proof.** Do Exercise 2.4.3.

Still, in the minimal logic we can prove (Brouwer, 1923?):

**Theorem 2.4.5.** [$L_1$, $L_2$, $L_9$, MP]: ⊢¬¬¬A↔¬A. What does it mean?

**Proof**. Indeed, by Theorem 2.4.4, ⊢¬A→¬¬¬A. By the Contraposition Law (Theorem 2.4.2), ⊢ (A→¬¬A)→(¬¬¬A→¬A). Hence, by Theorem 2.4.4, ⊢ ¬¬¬A→¬A. Q.E.D.

**Exercise 2.4.3**. Use Theorem 2.4.1 to prove Theorems 2.4.3 and 2.4.4.

Theorem 2.4.5 (and some of the following formulas in this and in the next section containing double negations) may seem uninteresting to people believing unconditionally in the equivalence ¬¬A↔A. Still, it seems interesting (at least – for a mathematician) to obtain a general characterization of logical formulas that do not depend on the Law of Excluded Middle. In Section 2.7 we will use these formulas to prove the elegant and non-trivial Glivenko's theorem: a) A is provable in the classical propositional logic (in [$L_1$-$L_{11}$, MP]), if and only if ¬¬A is provable in the constructive propositional logic (in [$L_1$-$L_{10}$, MP]), b) ¬A is provable in the classical propositional logic, if and only if ¬A is provable in the constructive propositional logic.

**Theorem 2.4.6.** a) [$L_1$, $L_2$, $L_9$, MP]: ⊢(¬A→A)→¬¬A. What does it mean?

b) [$L_1$, $L_2$, $L_6$, $L_7$, $L_9$, MP]: ⊢¬¬(Av¬A). What does it mean?

In this weak form, the Law of Excluded Middle can be "proved constructively". The formula $\neg\neg(A \vee \neg A)$ can be proved in the constructive logic, but $A \vee \neg A$ can't – as we will see in Section 2.8.

**Proof**. Do Exercise 2.4.4.

**Exercise 2.4.4.** Prove (a) and (b) of Theorem 2.4.6. The axiom $L_{11}$ must not be used in these proofs! (Hint for (b): use axioms to derive a contradiction from ¬(Av¬A).)

**Theorem 2.4.7.** [$L_1$-$L_9$, MP]: a) ⊢ (A→B)→(¬¬A→¬¬B). What does it mean?

b) ⊢¬¬(A→B)→(¬¬A→¬¬B). What does it mean?

c) ⊢(A→(B→C))→(¬¬A→(¬¬B→¬¬C)). What does it mean?

d) ¬¬(A→B), ¬¬(B→C) ⊢ ¬¬(A→C). What does it mean?

e) ¬¬A, ¬¬(A→B) ⊢ ¬¬B. What does it mean?

The converse of (a): (¬¬A→¬¬B)→(A→B) cannot be proved in the constructive logic (see Section 2.8).

**Proof**. To prove (a), we must simply apply twice the Contraposition Law: (A→B)→(¬B→¬A)→(¬¬A→¬¬B).

Now, let us prove (b).

| | | |
|---|---|---|
| (1) | ¬¬(A→B) | Hypothesis. |
| (2) | ¬¬A | Hypothesis. |
| (3) | ¬¬A→((A→B)→¬¬B) | From (a), by transposing A→B and ¬¬A, by the Premise Permutation Law. |
| (4) | (A→B)→¬¬B | From (2) and (3). |
| (5) | ((A→B)→¬¬B)→(¬¬¬B→¬(A→B)) | By the Contraposition Law. |
| (6) | ¬¬¬B→¬(A→B) | From (4) and (5). |
| (7) | (¬¬¬B→¬(A→B))→(¬¬(A→B)→¬¬¬¬B) | By the Contraposition Law. |
| (8) | ¬¬(A→B)→¬¬¬¬B | From (6) and (7). |
| (9) | ¬¬¬¬B | From (1) and (8). |
| (10) | ¬¬¬¬B→¬¬B | By Theorem 2.4.5. |
| (11) | ¬¬B | From (9) and (10). |

Thus, by Deduction Theorem 1, ⊢ ¬¬(A→B)→(¬¬A→¬¬B).

To prove (c) do Exercise 2.4.5.

To prove (d), first, let us take (c) with A = A→B, B = B→C, C = A→C:

(1)⊢ ((A→B)→((B→C)→(A→C)))→(¬¬(A→B)→(¬¬(B→C)→¬¬(A→C))).

| | | |
|---|---|---|
| (2) | ⊢ (A→B)→((B→C)→(A→C) | By transitivity of implication and Deduction Theorem 1. |

(3)  ¬¬(A→B)                    Hypothesis.

(4)  ¬¬(B→C)                    Hypothesis.

(5)  ¬¬(A→C)                    From (1), (3) and (4).

Of course, (e) is an easy consequence of (b). Q.E.D.

**Exercise 2.4.5**. Prove (c) of Theorem 2.4.7. (Hint: apply (a) and (b).)

**Theorem 2.4.8.** [$L_1$-$L_9$, MP]: a) $\vdash \neg\neg(A \wedge B) \leftrightarrow (\neg\neg A \wedge \neg\neg B)$ . What does it mean?

b) $\vdash \neg\neg A \vee \neg\neg B \rightarrow \neg\neg(A \vee B)$ . What does it mean?

**Proof**. Do Exercise 2.4.6.

**Attention: non-constructive reasoning!** The converse of (b): $\neg\neg(A \vee B) \rightarrow \neg\neg A \vee \neg\neg B$ cannot be proved in the constructive logic (see [Section 2.8](#)). What does it mean? If we simply succeed in deriving a contradiction from $\neg(A \vee B)$ , then, perhaps, we do not have a method allowing to decide, which part of $\neg\neg A \vee \neg\neg B$ is true – ¬¬A, or ¬¬B?

**Exercise 2.4.6**. a) Prove Theorem 2.4.8(a). (Hint: use axioms $L_3$–$L_5$, the Contraposition Law – several times, and Theorem 2.4.7(b).)

b) Prove Theorem 2.4.8(b). (Hint: use Theorem 2.3.1 and the Contraposition Law.)

**Theorem 2.4.9.** [$L_1$-$L_9$, MP]:

a) $\vdash \neg A \vee \neg B \rightarrow \neg(A \wedge B)$ . It's a **half of** the so-called **First de Morgan Law**. What does it mean?

b) $\vdash \neg(A \vee B) \leftrightarrow \neg A \wedge \neg B$ . It's the so-called **Second de Morgan Law**. What does it mean?

**Proof**. Do Exercise 2.4.7.

**Attention: non-constructive reasoning!** The second half of (a) – the converse implication, i.e. the equivalence $\neg(A \wedge B) \leftrightarrow \neg A \vee \neg B$ can be proved in the classical logic, yet not in the constructive logic (see [Section 2.8](#)). Explain, why.

[Augustus de Morgan](#) (1806-1871): "He recognised the purely symbolic nature of algebra and he was aware of the existence of algebras other than ordinary algebra. He introduced de Morgan's laws and his greatest contribution is as a reformer of mathematical logic." (according to [MacTutor History of Mathematics archive](#)).

**Exercise 2.4.7.** Prove:

a) Prove (a) of Theorem 2.4.9. (Hint: apply Theorems 2.3.1 and 2.4.1).

b) Prove (b) of Theorem 2.4.9. (Hint: ($\leftarrow$) – apply Theorem 2.4.1, ($\rightarrow$) – apply the Contraposition Law).

c) [$L_1$-$L_9$, MP]: $\vdash (A \rightarrow B) \rightarrow \neg(A \wedge \neg B)$ . What does it mean? Compare with Exercise 2.4.1.

d) [$L_1$-$L_8$, MP]: $\vdash A \vee B \rightarrow ((A \rightarrow B) \rightarrow B)$ . What does it mean?

**Attention: non-constructive reasoning!** The converse implication of (a), $\neg(A \wedge \neg B) \rightarrow (A \rightarrow B)$ cannot be proved in the constructive logic (see Section 2.8). Explain, why. Still, we will prove this formula in the classical logic.

The converse of (b): $((A \rightarrow B) \rightarrow B) \rightarrow A \vee B$ cannot be proved in the constructive logic (see Section 2.8). Explain, why. Still, we will prove this formula in the classical logic.

## 2.5. Formulas Containing Negation – Constructive Logic

In this book, **constructive logic** is used as a synonym of **intuitionistic logic**!

Constructive logic includes the "crazy" axiom $\mathbf{L_{10}}$: $\neg B \rightarrow (B \rightarrow C)$, but rejects the Law of Excluded Middle $\mathbf{L_{11}}$: $B \vee \neg B$ as a general logical principle.

**Theorem 2.5.1.** a) [$L_{10}$, MP]: A, $\neg A \vdash B$. What does it mean?

b) [$L_1$, $L_2$, $L_8$, $L_{10}$, MP]: $\vdash A \vee B \rightarrow (\neg A \rightarrow B)$ . What does it mean?

c) [$L_1$, $L_8$, $L_{10}$, MP]: $\vdash \neg A \vee B \rightarrow (A \rightarrow B)$ . What does it mean?

d) [$L_1$, $L_2$, $L_6$, MP]: $A \vee B \rightarrow (\neg A \rightarrow B) \vdash \neg A \rightarrow (A \rightarrow B)$. What does it mean? The "natural" rule $A \vee B, \neg A \vdash B$ implies $L_{10}$!

**Attention: non-constructive reasoning!** The converse of (b), i.e. $(\neg A \rightarrow B) \rightarrow A \vee B$ cannot be proved in the constructive logic (see Section 2.8). Explain, why. The converse of (c), i.e. $(A \rightarrow B) \rightarrow \neg A \vee B$ cannot be proved in constructive logic as well (see Section 2.8). Explain, why.

**Proof.** Of course, (a) follows directly from $L_{10}$, by MP.

To prove (b) and (c) do Exercise 2.5.1.

Let us prove (d). Surprisingly, the rule $A \vee B, \neg A \vdash B$ seems to be a quite a "natural" logical principle, yet it cannot be proved without the "crazy" axiom $L_{10}$! Why not? Because it implies $L_{10}$! Indeed,

(1)   $A \vee B \rightarrow (\neg A \rightarrow B)$     Hypothesis.

(2)  ¬A     Hypothesis.

(3)  A     Hypothesis.

(4)   $A \rightarrow A \vee B$     Axiom $L_6$.

(5)   $A \vee B$     By MP, from (3) and (4).

(6)  B     By MP, from (1), (5) and (2).

Hence, by Deduction Theorem 1, $[L_1, L_2, L_6, MP]$:  $A \vee B \rightarrow (\neg A \rightarrow B)$  ⊢ ¬A→(A→B).

In <u>Section 2.8</u> we will prove that $L_{10}$ cannot be derived from $L_1$-$L_9$, hence, (b) also cannot be derived from $L_1$-$L_9$ (i.e. without $L_{10}$). Q.E.D.

**Exercise 2.5.1.** Prove (b) and (c) of Theorem 2.5.1. Note: when proving (c), you cannot use Deduction Theorem 1 (because of the missing axiom $L_2$). So, simply build a sequence of 5 formulas representing the proof of (c) as in Exercise 1.4.2.

**Theorem 2.5.2.** $[L_1$-$L_{10}, MP]$:

a) ⊢ (¬¬A→¬¬B)→¬¬(A→B). It's the converse of Theorem 2.4.7(b). Hence, $[L_1$-$L_{10}, MP]$: ⊢¬¬(A→B)↔(¬¬A→¬¬B).

b) ⊢¬¬A→(¬A→A). It's the converse of Theorem 2.4.6(a). Hence,

$[L_1$-$L_{10}, MP]$: ⊢¬¬A↔(¬A→A).

c) ⊢  $A \vee \neg A \rightarrow (\neg\neg A \rightarrow A)$  . What does it mean?

d) ⊢¬¬(¬¬A→A). What does it mean?

**Proof**. Of course, (b) is an instance of the **axiom $L_{10}$**.

To prove (a), prove in Exercise 2.5.2 that ¬¬A→¬¬B, ¬(A→B) ⊢¬B, ¬¬B. Then, by Theorem 2.4.1, ⊢(¬¬A→¬¬B)→¬¬(A→B).

To prove (c) and (d) do Exercise 2.5.2. Q.E.D.

**Exercise 2.5.2**. a) Prove that $[L_1$-$L_{10}, MP]$: ¬¬A→¬¬B, ¬(A→B) ⊢¬B, ¬¬B.

b) Prove (c) and (d) of Theorem 2.5.2.

**Exercise 2.5.3.** Prove that in $[L_1$-$L_{10}, MP]$:

a) A ⊢  $B \leftrightarrow B \vee \neg A$   . What does it mean?

b) $\vdash\ B\vee\left(A\wedge\neg A\right)\leftrightarrow B$ . What does it mean?

c) $\vdash\ \left(\left(A\wedge\neg A\right)\wedge B\right)\vee C\leftrightarrow C$ . What does it mean?

## 2.6. Formulas Containing Negation – Classical Logic

If you agree to adopt the formula $B\vee\neg B$ , i.e. the Law of Excluded Middle (axiom $L_{11}$ in the list of Section 1.3), you can prove, first of all, the so called **Double Negation Law**:

**Theorem 2.6.1.** [$L_1$-$L_{11}$, MP]: $\vdash\neg\neg A \to A$. Hence, [$L_1$-$L_{11}$, MP]: $\vdash\neg\neg A \leftrightarrow$ A.

**Proof.** Indeed, by Theorem 2.5.2(c),

$$[L_1\text{-}L_{10}, \text{MP}]: \vdash\ A\vee\neg A\to\left(\neg\neg A\to A\right)\ ,$$

hence, [$L_1$-$L_{11}$, MP]: $\vdash\neg\neg A\to A$. Q.E.D.

In the minimal logic we proved Theorem 2.4.4: [$L_1$, $L_2$, $L_9$, MP]: $\vdash A\to\neg\neg A$. Hence, [$L_1$-$L_{11}$, MP]: $\vdash\neg\neg A \leftrightarrow A$.

**Attention: non-constructive reasoning!** The formula $\neg\neg A\to A$ cannot be proved in the constructive logic, see Section 2.8. Why? Because it represents a kind of non-constructive reasoning. Indeed, imagine, you wish to prove that $\exists xB(x)$. Assume the contrary, $\neg\exists xB(x)$, and derive a contradiction. Thus you have proved... the negation of $\neg\exists xB(x)$, i.e. $\neg\neg\exists xB(x)$. To conclude $\exists xB(x)$ from $\neg\neg\exists xB(x)$, you need the Double Negation Law. Hence, by adopting this law as a logical principle, you would allow non-constructive existence proofs – if you prove $\exists xB(x)$ by assuming $\neg\exists xB(x)$, and deriving a contradiction, then you may not obtain a method allowing to find a particular x satisfying B(x).

**Theorem 2.6.1A.** [$L_8$, $L_{11}$, MP]: A→B, ¬A→B $\vdash$ B. Or, by Deduction Theorem 1, [$L_1$, $L_2$, $L_8$, $L_{11}$, MP]: (A→B)→((¬A→B)→B). What does it mean?

This formula cannot be proved in the constructive logic (see Section 2.8). Explain, why.

**Exercise 2.6.1.** Prove Theorem 2.6.1A.

In the classical logic, you can prove also the full form of the **Contraposition Law:**

**Theorem 2.6.2.** [L$_1$-L$_{11}$, MP]: ⊢ (¬B→¬A)→(A→B). Hence, [L$_1$-L$_{11}$, MP]: ⊢(A→B) ↔ (¬B→¬A).

**Attention: non-constructive reasoning!** The formula (¬B→¬A) → (A→B) cannot be proved in the constructive logic, see Section 2.8. Explain, why.

**Exercise 2.6.2.** Prove that in [L$_1$-L$_{11}$, MP]:

a) Theorem 2.6.2. (Hint: apply Double Negation Law.)

b) ⊢(¬A→B)↔(¬B→A) (compare with Theorem 2.4.3).

**Attention: non-constructive reasoning!** These formulas cannot be proved in the constructive logic, see Section 2.8.

**Theorem 2.6.3.** [L$_1$-L$_{11}$, MP]: ⊢ $\neg(A \wedge B) \leftrightarrow \neg A \vee \neg B$ . It's the so-called **First de Morgan Law.**

**Attention: non-constructive reasoning!** This formula cannot be proved in the constructive logic, see Section 2.8. Explain, why.

**Proof**. The first half of this Law we proved in the minimal logic as Theorem 2.4.9(a): [L$_1$-L$_9$, MP] ⊢ $\neg A \vee \neg B \rightarrow \neg(A \wedge B)$ . Let us prove the remaining half: [L$_1$-L$_{11}$, MP] ⊢ $\neg(A \wedge B) \rightarrow \neg A \vee \neg B$ .

Let us start by proving $\neg(\neg A \vee \neg B) \rightarrow A \wedge B$ .

| (1) | $\neg(\neg A \vee \neg B)$ | Hypothesis. |
|---|---|---|
| (2) | $\neg(\neg A v \neg B) \rightarrow \neg\neg A \wedge \neg\neg B$ | By the Second de Morgan Law – Theorem 2.4.9(b). |
| (3) | $\neg\neg A \wedge \neg\neg B ; \neg\neg A ; \neg\neg B$ | MP, etc. |
| (4) | $A ; B ; A \wedge B$ | Double Negation Law, etc. |

Hence, by Deduction Theorem 1, [L$_1$-L$_{11}$, MP] ⊢ $\neg(\neg A \vee \neg B) \rightarrow A \wedge B$ . By applying the first half of Contraposition Law [L$_1$-L$_{11}$, MP] ⊢ $\neg(A \wedge B) \rightarrow \neg\neg(\neg A \vee \neg B)$ . By Double Negation Law, [L$_1$-L$_{11}$, MP] ⊢ $\neg\neg(\neg A \vee \neg B) \rightarrow \neg A \vee \neg B$ , hence, [L$_1$-L$_{11}$, MP] ⊢ $\neg(A \wedge B) \rightarrow \neg A \vee \neg B$ . Q.E.D.

In the classical logic, we can express implication by using negation and disjunction. Indeed, we already know that [L$_1$-L$_{10}$, MP]: ⊢ $\neg A \vee B \rightarrow (A \rightarrow B)$ (Theorem 2.5.1(c)).

**Theorem 2.6.4.** a) [L$_1$-L$_8$, MP]: $A \vee C$ ⊢ $(A \rightarrow B) \rightarrow B \vee C$ . Hence, [L$_1$-

$L_8$, MP]:   $A \vee \neg A \;\vdash\; (A \to B) \to \neg A \vee B$  .

b) [$L_1$-$L_{11}$, MP]: $\vdash\; (A \to B) \leftrightarrow \neg A \vee B$  .

**Proof**. To prove (a), do Exercise 2.6.3.

And, of course, (b) follows from (a) and Theorem 2.5.1(c). Q.E.D.

**Exercise 2.6.3**. a) Use Theorem 2.3.1 to prove (a) of Theorem 2.6.4.

Prove that in [$L_1$-$L_{11}$, MP]:

b) $\vdash\; B \wedge (A \vee \neg A) \leftrightarrow B$  . What does it mean?

c) $\vdash\; ((A \vee \neg A) \vee B) \wedge C \leftrightarrow C$  . What does it mean?

d) $\vdash\; ((A \to B) \to B) \to A \vee B$  . What does it mean? Hence, by adding the result of Exercise 2.4.7(d), [$L_1$-$L_{11}$, MP]: $\vdash\; A \vee B \leftrightarrow ((A \to B) \to B)$  .

Thus, in the classical logic, disjunctions can be replaced by implications.

**Exercise 2.6.4**. Prove that in [$L_1$-$L_{11}$, MP]:

a) $\vdash\; (A \to B) \leftrightarrow \neg(A \wedge \neg B)$  . What does it mean?

b) $\vdash\; \neg(A \to B) \leftrightarrow A \wedge \neg B$  . What does it mean?

c) $\vdash\; A \vee B \leftrightarrow (\neg A \to B)$  . What does it mean?

d) $\vdash\; A \wedge B \leftrightarrow \neg(A \to \neg B)$  . What does it mean?

e) (optional, for smart students) Try detecting, which parts of these equivalences are provable in the constructive logic.


**Strange formulas**

**Exercise 2.6.5**. Prove in the classical logic the following strange formulas:

a) [$L_1$, $L_2$, $L_6$–$L_8$, $L_{10}$, $L_{11}$, MP] $\vdash\; A \vee (A \to B)$  . What does it mean? Does it mean anything at all? Compare with Exercise 1.4.2 (i).

b) [$L_1$, $L_2$, $L_6$–$L_8$, $L_{10}$, $L_{11}$, MP] $\vdash\; (A \to B) \vee (B \to A)$  . What does it mean? Does it mean anything at all? The most crazy theorem of the classical propositional logic?

c) [$L_1$–$L_{11}$, MP] $\vdash\; ((A \to B) \to A) \to A$  . What does it mean? Does it mean anything at all? It is the so-called **Peirce's Law** from:

**C. S. Peirce**. On the algebra of logic: A contribution to the philosophy of notation. *American Journal of Mathematics*, 1885, vol.7, pp.180-202.

## 2.7. Constructive Embedding. Glivenko's Theorem

Let us remind some of the results of previous sections concerning double negations:

**Theorem 2.4.4.** [$L_1$, $L_2$, $L_9$, MP]: $\vdash A \to \neg\neg A$.

**Theorem 2.4.5.** [$L_1$–$L_9$, MP]: $\vdash \neg\neg\neg A \leftrightarrow \neg A$.

**Theorem 2.4.6(b).** [$L_1$–$L_9$, MP]: $\vdash \neg\neg(A \vee \neg A)$. In this weak form, the Law of Excluded Middle can be "proved constructively".

**Theorem 2.4.7.** [$L_1$–$L_9$, MP]: a) $\vdash (A \to B) \to (\neg\neg A \to \neg\neg B)$.

b) $\vdash \neg\neg(A \to B) \to (\neg\neg A \to \neg\neg B)$.

c) $\vdash (A \to (B \to C)) \to (\neg\neg A \to (\neg\neg B \to \neg\neg C))$.

d) $\neg\neg(A \to B)$, $\neg\neg(B \to C) \vdash \neg\neg(A \to C)$.

e) $\neg\neg A$, $\neg\neg(A \to B) \vdash \neg\neg B$.

**Theorem 2.4.8.** [$L_1$-$L_9$, MP]: a) $\vdash \neg\neg(A \wedge B) \leftrightarrow (\neg\neg A \wedge \neg\neg B)$.

b) $\vdash \neg\neg A \vee \neg\neg B \to \neg\neg(A \vee B)$.

**Theorem 2.5.2.** [$L_1$-$L_{10}$, MP]: a) $\vdash (\neg\neg A \to \neg\neg B) \to \neg\neg(A \to B)$. It's the converse of Theorem 2.4.7(b).

d) $\vdash \neg\neg(\neg\neg A \to A)$.

**Theorem 2.6.1.** [$L_1$-$L_{11}$, MP]: $\vdash \neg\neg A \leftrightarrow A$.

Does it mean that for **any** formula A: if [$L_1$-$L_{11}$, MP]: $\vdash A$, then [$L_1$-$L_{10}$, MP]: $\vdash \neg\neg A$? (The converse is obvious: if [$L_1$-$L_{10}$, MP]: $\vdash \neg\neg A$, then [$L_1$-$L_{11}$, MP]: $\vdash A$ by Theorem 2.6.1.)

Imagine, we have a proof of [$L_1$-$L_{11}$, MP]: $\vdash A$. It is a sequence of formulas $R_1$, $R_2$, ..., $R_n$, where $R_n = A$. If this sequence does not contain instances of the axiom $L_{11}$, then it is a proof of [$L_1$-$L_{10}$, MP]: $\vdash A$ as well. Hence, according to Theorem 2.4.4, [$L_1$-$L_{10}$, MP]: $\vdash \neg\neg A$

If the sequence $R_1$, $R_2$, ..., $R_n$ contains some instances of $L_{11}$, i.e. formulas having the form $B \vee \neg B$, then, according to Theorem 2.4.6(b), we could try replacing each such formula by a sequence proving that [$L_1$-$L_9$, MP]: $\vdash \neg\neg(B \vee \neg B)$. It appears that **each** of the formulas $\neg\neg R_1$, $\neg\neg R_2$, ..., $\neg\neg R_n$ is

provable in $[L_1\text{-}L_{10}, MP]$.

a) If $R_k$ is an instance of the axioms $L_1\text{-}L_{10}$, then $[L_1\text{-}L_{10}, MP]: \vdash \neg\neg R_k$ (Theorem 2.4.4).

b) If $R_k$ is an instance of the axiom $L_{11}$, then $[L_1\text{-}L_{10}, MP]: \vdash \neg\neg R_k$ (Theorem 2.4.6(b)).

c) Now, let us assume that i, j < k, and $R_i$, $R_j \vdash R_k$ directly by MP, i.e. $R_j$ is $R_i \rightarrow R_k$. We know already that $[L_1\text{-}L_{10}, MP]: \vdash \neg\neg R_i$ and $[L_1\text{-}L_{10}, MP]: \vdash \neg\neg(R_i \rightarrow R_k)$. By Theorem 2.4.7(b), $[L_1\text{-}L_9, MP]: \vdash \neg\neg(R_i \rightarrow R_k) \rightarrow (\neg\neg R_i \rightarrow \neg\neg R_k)$. Hence, $[L_1\text{-}L_{10}, MP]: \vdash \neg\neg R_k$.

Because $A = R_n$, we have proved the remarkable Glivenko's theorem from 1929:

**V.Glivenko**. Sur quelques points de la logique de M. Brouwer. *Academie Royale de Belgique, Bulletins de la classe des sciences*, 1929, ser.5, vol.15, pp.183-188.

Valery Ivanovich Glivenko (1897-1940, photo) is best known by the so-called Glivenko-Cantelli theorem in probability theory.

**Glivenko's Theorem**. $[L_1\text{-}L_{11}, MP]: \vdash A$, if and only if $[L_1\text{-}L_{10}, MP]: \vdash \neg\neg A$.

Or: a formula A is provable in the classical propositional logic, if and only if its double negation $\neg\neg A$ is provable in the constructive propositional logic.

This theorem provides a kind of a "constructive embedding" for the classical propositional logic: any classically provable formula can be "proved" in the constructive logic, if you simply put two negations before it.

**Corollary**. $[L_1\text{-}L_{11}, MP]: \vdash \neg A$, if and only if $[L_1\text{-}L_{10}, MP]: \vdash \neg A$. Or: a "negative" formula $\neg A$ is provable in the classical propositional logic, if and only if it is provable in the constructive propositional logic.

Indeed, if $[L_1\text{-}L_{11}, MP]: \vdash \neg A$, then by Glivenko's theorem, $[L_1\text{-}L_{10}, MP]: \vdash \neg\neg\neg A$, and by Theorem 2.4.5, $[L_1\text{-}L_{10}, MP]: \vdash \neg A$. Q.E.D.

**Exercise 2.7.1.** Prove the following version of Glivenko's theorem (see Kleene [1952]):

a) If $[L_1\text{-}L_{11}, MP]: A_1, A_2, ..., A_n \vdash C$, then

$[L_1\text{-}L_{10}, MP]: \neg\neg A_1, \neg\neg A_2, ..., \neg\neg A_n \vdash \neg\neg C$.

b) If $[L_1\text{-}L_{11}, MP]: \neg A_1, \neg A_2, ..., \neg A_n, B_1, B_2, ..., B_p \vdash \neg C$, then

$[L_1\text{-}L_{10}, MP]: \neg A_1, \neg A_2, ..., \neg A_n, \neg\neg B_1, \neg\neg B_2, ..., \neg\neg B_p \vdash \neg C$.

## 2.8. Axiom Independence. Using Computers in Mathematical Proofs

If one of our axioms $L_i$ could be proved by using the other axioms only, then we could simplify our logical system by dropping $L_i$ as an axiom. A striking example:

**Theorem 2.8.1**. The axiom **$L_9$**: $(A{\to}B){\to}((A{\to}\neg B){\to}\neg A)$ can be proved in $[L_1, L_2, L_8, L_{10}, L_{11}, MP]$.

This fact was established by Augusts Kurmitis (on the web, also: A. A. Kurmit):

**A. A. Kurmitis**. On independence of a certain axiom system of the propositional calculus. *Proc. Latvian State University*, 1958, Vol. 20, N3, pp. 21-25 (in Russian).

The following proof of $L_9$ in $[L_1, L_2, L_8, L_{10}, L_{11}, MP]$ is due to Janis Sedols (1939-2011).

First, let us establish that the formula $(A{\to}\neg A){\to}\neg A$ can be proved in $[L_1, L_2, L_8, L_{10}, L_{11}, MP]$ (in Exercise 2.4.1 we established that $[L_1, L_2, L_9, MP]$: $\vdash (A{\to}\neg A){\to}\neg A)$:

| | | |
|---|---|---|
| (1) | $(A \to \neg A) \to ((\neg A \to \neg A) \to (A \lor \neg A) \to \neg A)$ | Axiom $L_8$. |
| (2) | $A{\to}\neg A$ | Hypothesis. |
| (3) | $\neg A{\to}\neg A$ | This is provable in $[L_1, L_2, MP]$ (Theorem 1.4.1). |
| (4) | $A \lor \neg A$ | Axiom $L_{11}$. |
| (4) | $\neg A$ | From (1), (2), (3) and (4), by MP. |
| (6) | $(A{\to}\neg A){\to}\neg A$ | By Deduction Theorem 1 (which is valid for any logical system containing $[L_1, L_2, MP]$). |

Now let us establish that in $[L_1, L_2, L_{10}, MP]$: $A{\to}B, A{\to}\neg B \vdash A{\to}\neg A$.

| | | |
|---|---|---|
| (7) | $A{\to}B$ | Hypothesis. |

| | | |
|---|---|---|
| (8) | A→¬B | Hypothesis. |
| (9) | A | Hypothesis. |
| (9) | B | From (7), (9), by MP. |
| (10) | ¬B | From (8), (9), by MP. |
| (11) | ¬B→(B→¬A) | Axiom $L_{10}$. |
| (12) | ¬A | From (9), (10) and (11) by MP. |
| (13) | A→B, A→¬B ⊢A→¬A | By Deduction Theorem 1 (which is valid for any propositional system containing [$L_1$, $L_2$, MP]). |

Finally, let us merge the proofs of (6) and (13), then by MP we obtain ¬A, i.e.

[$L_1$, $L_2$, $L_8$, $L_{10}$, $L_{11}$, MP]: A→B, A→¬B ⊢¬A.

Now, by Deduction Theorem 1 (which is valid for any propositional system containing [$L_1$, $L_2$, MP]) we obtain the axiom $L_9$:

$$[L_1, L_2, L_8, L_{10}, L_{11}, \text{MP}]: (A{\to}B){\to}((A{\to}\neg B){\to}\neg A).$$

Q.E.D.

What should we do after establishing that one of our axioms is "dependent"? Do you think, we should drop $L_9$ as an axiom of our logical system?

First, let's note that **we have proved $L_9$ by using three problematic axioms**: $L_1$, $L_{10}$, $L_{11}$. But $L_9$ itself is not problematic!

Secondly, $L_9$ **cannot** be proved in [$L_1$-$L_8$, $L_{10}$, MP] (see Theorem 2.8.2 below). Hence, if we would drop $L_9$, then, instead of a simple definition

$$\text{classical logic} = \text{constructive logic} + L_{11},$$

we would have a more complicated one:

$$\text{constructive logic} = \text{classical logic} - L_{11} + L_9.$$

Now, the question of questions:

**Is the Law of Excluded Middle an independent logical principle?**

Could we prove the Law of Excluded Middle (the axiom $L_{11}$: $B \lor \neg B$ ) by using the other axioms [$L_1$-$L_{10}$, MP] as we proved $L_9$ in [$L_1$, $L_2$, $L_8$, $L_{10}$, $L_{11}$,

MP]? If not, how could we demonstrate that this is impossible at all? How could we demonstrate that some logical principle is **independent**, i.e. that it cannot be derived from other principles?

Let us assume, we have designed an algorithm $q$ that calculates for each formula A some its "property" q(A) such that:

a) $q(L_1)$ is true, $q(L_2)$ is true, ..., $q(L_{10})$ is true (i.e. the axioms $L_1$-$L_{10}$ possess the property $q$).

b) If q(A) is true and q(A→B) is true, then q(B) is true (i.e. MP "preserves" the property q).

Then q(F) is true for all the formulas F that are provable in $[L_1$-$L_{10}$, MP].

c) $q(L_{11})$ is false ($L_{11}$ does not possess the property $q$).

If we could obtain such a property $q$, then, of course, this would demonstrate that $L_{11}$ cannot be proved in $[L_1$-$L_{10}$, MP], i.e. that the **Law of Excluded Middle is an independent logical principle**.

The most popular way of introducing such properties of formulas are the so-called "multi-valued logics" or "many-valued logics", introduced by Jan Lukasiewicz and Emil Post:

**J.Lukasiewicz.** O logice trojwartosciowej. *Ruch Filozoficzny (Lwow)*, 1920, vol. 5, pp. 169-171

**E.Post**. Introduction to a general theory of elementary propositions. *Amer. journ. math.*, 1921, vol. 21, pp.163-195

Read more: Many-Valued Logic by **Siegfried Gottwald** in Stanford Encyclopedia of Philosophy.

For example, let us consider a kind of "three-valued logic", where 0 means "false", 1 – "unknown" (or NULL – in terms of SQL), 2 – "true". Then it would be natural to define conjunction and disjunction as

$$A \wedge B = \min(A,B)$$
$$A \vee B = \max(A,B).$$

But how should we define implication and negation?

| A | B | $A \wedge B$ | $A \vee B$ | A→B |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | $i_1$ |
| 0 | 1 | 0 | 1 | $i_2$ |
| 0 | 2 | 0 | 2 | $i_3$ |

| 1 | 0 | 0 | 1 | $i_4$ |
|---|---|---|---|-------|
| 1 | 1 | 1 | 1 | $i_5$ |
| 1 | 2 | 1 | 2 | $i_6$ |
| 2 | 0 | 0 | 2 | $i_7$ |
| 2 | 1 | 1 | 2 | $i_8$ |
| 2 | 2 | 2 | 2 | $i_9$ |

| A | ¬A |
|---|-----|
| 0 | $i_{10}$ |
| 1 | $i_{11}$ |
| 2 | $i_{12}$ |

Thus, theoretically, we have here to explore: $3^9 = 19683$ variants of implication definitions and $3^3 = 27$ negation definitions.

Do you think, it would be natural to set the values of ¬A and A→B as follows?

| A | ¬A |
|---|-----|
| 0 | 2 |
| 1 | 1 |
| 2 | 0 |

| A | B | A→B |
|---|---|-----|
| 0 | 0 | 2 |
| 0 | 1 | 2 |
| 0 | 2 | 2 |
| 1 | 0 | 1 |

| | | |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 2 | 2 |
| 2 | 0 | 0 |
| 2 | 1 | 1 |
| 2 | 2 | 2 |

Yes, if we would try building a "natural" three-valued logic, in which "1" would mean, indeed, "unknown". In this way we would obtain the "natural" three-valued logic used, for example, for handling of <u>NULL-values in SQL</u>.

However, for our purpose, this logic is useless: none of the axioms $L_1$- $L_{11}$ always take "true" values (i.e. 2). Indeed, if the atoms B, C, D all take the value 1, then the axioms also take the value 1. For example:

| B | ¬B | $B \lor \neg B$ |
|---|---|---|
| 0 | 2 | 2 |
| 1 | 1 | 1 |
| 2 | 0 | 2 |

Thus, our aim is here must the opposite of the "natural". We will consider

"under the above truth tables, formula A always takes "true" values"

as a kind of the above-mentioned "property" q(A). Hence, we will try to define the tables for implication and negation in such a way that:

a) the axioms $L_1$, $L_2$, ..., $L_{10}$ always take "true" values (i.e. 2),

b) MP preserves taking always "true" values (i.e. if the formulas A and A→B are always 2, then B also is always 2),

c) the axiom $L_{11}$ sometimes takes the values 0 or 1.

As we saw above, the definitions of implication and negation, having these properties, cannot be 100% natural. So, we must explore (at least some of) the "unnatural" versions as well.

**Exercise 2.8.1** (optional). Develop a simple (recursive) computer program receiving as input:

a) Any such "truth tables".

b) Any formula F consisting of letters A, B, C, and propositional connectives.

and printing out "truth values" of the formula F, for example, if F = B→(A→B):

| A | B | B→(A→B) |
|---|---|---------|
| 0 | 0 | 2 |
| 0 | 1 | 2 |
| 0 | 2 | 2 |
| 1 | 0 | 2 |
| 1 | 1 | 2 |
| 1 | 2 | 2 |
| 2 | 0 | 2 |
| 2 | 1 | 2 |
| 2 | 2 | 2 |

In this example the axiom $L_1$ always takes "true" values. Perhaps, we should be interested only in those variants of our "truth tables" that "satisfy" at least the axioms $L_1$, $L_2$, ..., $L_8$ forcing them always to take "true" values.

**Note**. See my version of the program in C++: header file, implementation.

Thus, we consider

"under the above truth tables, formula A always takes "true" values"

as a kind of the "property" q(A).

Will MP preserve this property? If A is "true", and A→B is "true", how could B be not? Let us consider the relevant part of the above truth tables (i.e. the part where A is "true"):

| A | B | A→B |
|---|---|-----|
| 2 | 0 | $i_7$ |
| 2 | 1 | $i_8$ |
| 2 | 2 | $i_9$ |

If we would consider only those variants of our "truth tables" where $i_7 = 0$ or 1, $i_8 = 0$ or 1, and $i_9 = 2$, then, if B would not be 2 for some values of its arguments, then A→B also would not be 2 for the same values of arguments.

Hence, if we restrict ourselves to "truth tables" with $i_7 = 0$ or 1, $i_8 = 0$ or 1, and

$i_9$ = 2, then MP preserves the property of "being true". **I.e., from "true" formulas MP can derive only "true" formulas.**

The next idea: if we wish the axiom $L_6$: A→AvB always taking the value "true" (i.e. the value 2), then, **if A≤B, then A→B must be 2.**

Thus, of all the $3^9$ = 19683 possible implication definition variants only the following 3*2*2 = 12 variants are worth of exploring:

| A | B | A→B |
|---|---|-----|
| 0 | 0 | 2 |
| 0 | 1 | 2 |
| 0 | 2 | 2 |
| 1 | 0 | $i_4$=0,1,2 |
| 1 | 1 | 2 |
| 1 | 2 | 2 |
| 2 | 0 | $i_7$=0,1 |
| 2 | 1 | $i_8$=0,1 |
| 2 | 2 | 2 |

**Exercise 2.8.2**. a) Verify that under any of these 12 implication definitions the axioms $L_3$, $L_4$, $L_6$, $L_7$ always take the value 2, i.e. you do not need testing these axioms any more.

b) For each of the axioms, $L_1$, $L_2$, $L_5$ and $L_8$, determine all the possible combinations of the values of $i_4$, $i_7$, $i_8$ forcing it to take always the value 2.

**Note.** The "intersection" of b) consists of 5 combinations (see the results file #00).

**Exercise 2.8.3** (optional) Extend your previous computer program by adding 6 nested loops: for $i_4$=0 to 2, for $i_7$=0 to 1, for $i_8$=0 to1, for $i_a$=0 to 2, for $i_b$=0 to 2, for $i_c$=0 to 2. Let the program print out only those variants of "truth tables" that make "true" all the axioms $L_1$-$L_8$. (My program yields 135 such variants, see the results file #00).

Thus, now we have 135 variants of "truth tables" that make "true" all the

axioms $L_1$-$L_8$. Let us search among them for the variants that allow proving of axiom independency results we are interested in.

**Axiom L$_9$**

In Theorem 2.8.1 we established that the axiom **L$_9$**: (A→B)→((A→¬B)→¬A) can be proved in [$L_1$-$L_8$, $L_{10}$, $L_{11}$, MP]. Still,

**Theorem 2.8.2**. The axiom L$_9$ **cannot** be proved in [$L_1$-$L_8$, $L_{10}$, MP].

**Proof**. Let your program print out only those variants of "truth tables" that make "true" all the axioms $L_1$-$L_8$, and make: L$_9$ – not "true", and $L_{10}$ – "true".

My program yields 66 such variants, see the results file #01. Let us consider, for example, the variant #33 (the unnatural clauses are marked bold):

Implication variant #3:
2 2 2 **2** 2 2 0 1 2 L1-L8 true.
Variant #33. Negation: 2 1 0 L9 not true. L10 true. L11 not true.

| A | B | A→B |
|---|---|-----|
| 0 | 0 | 2 |
| 0 | 1 | 2 |
| 0 | 2 | 2 |
| **1** | **0** | **2** |
| **1** | **1** | **2** |
| 1 | 2 | 2 |
| 2 | 0 | 0 |
| 2 | 1 | 1 |
| 2 | 2 | 2 |

| A | ¬A |
|---|----|
| 0 | 2 |
| 1 | 1 |
| 2 | 0 |

See the extended results file #1 for this variant.

Under this variant the axioms $L_1$-$L_8$ and $L_{10}$ are "true". As we know, under this variant, by MP, from "true" formulas only "true" formulas can be derived. The axiom $L_9$ is not "true" under this variant:

| A | B | $(A{\to}B){\to}((A{\to}\neg B){\to}\neg A)$ |
|---|---|---|
| 0 | 0 | 2 |
| 0 | 1 | 2 |
| 0 | 2 | 2 |
| **1** | **0** | **1** |
| **1** | **1** | **1** |
| **1** | **2** | **1** |
| 2 | 0 | 2 |
| 2 | 1 | 2 |
| 2 | 2 | 2 |

Hence, $L_9$ **cannot** be proved in [$L_1$-$L_8$, $L_{10}$, MP]. Q.E.D.

In a similar way, we can obtain other independence results.

**Axiom $L_{10}$**

**Theorem 2.8.3**. The "crazy" axiom **$L_{10}$**: $\neg B{\to}(B{\to}C)$ cannot be proved in the minimal logic [$L_1$-$L_9$, MP], and even not in [$L_1$-$L_9$, $L_{11}$, MP].

**Proof**. Let your program print out only those variants of "truth tables" that make "true" all the axioms $L_1$-$L_8$, and make: $L_9$ – "true", $L_{10}$ – not "true", and $L_{11}$ – "true". My program yields 6 such variants, see the results file #02. Let us consider, for example, the variant #1 (the unnatural clauses are marked bold):

Implication variant #1:
2 2 2 **0 2** 2 0 1 2 L1-L8 true.
Variant #1. Negation: 2 **2** 1 L9 true. L10 not true. L11 true.

See the extended results file #2 for this variant.

Under this variant the axioms $L_1$-$L_9$ and $L_{11}$ are "true". As we know, under this variant, by MP, from "true" formulas only "true" formulas can be derived.

The axiom $L_{10}$ is not "true" under this variant:

| A | B | ¬A→(A→B) |
|---|---|----------|
| 0 | 0 | 2 |
| 0 | 1 | 2 |
| 0 | 2 | 2 |
| 1 | 0 | 2 |
| 1 | 1 | 2 |
| 1 | 2 | 2 |
| **2** | **0** | **0** |
| **2** | **1** | **1** |
| 2 | 2 | 2 |

Hence, $L_{10}$ **cannot** be proved in [$L_1$-$L_9$, $L_{11}$, MP]. Q.E.D.

**Axiom $L_{11}$**

Now, let us prove the main result of this section:

**Theorem 2.8.4.** The Law of Excluded Middle **$L_{11}$**: $B \lor \neg B$ cannot be proved in the constructive propositional logic [$L_1$-$L_{10}$, MP]. I.e. the **Law of Excluded Middle is an independent logical principle**.

**Proof**. Let your program print out only those variants of "truth tables" that make "true" all the axioms $L_1$-$L_8$, and make: $L_9$ – "true", $L_{10}$ – "true", $L_{11}$ – not "true". My program yields only one such variant, see the <u>results file #03</u>:

Implication variant #1(the unnatural clauses are marked bold):
2 2 2 **0 2** 2 0 1 2 L1-L8 true.
Variant #1. Negation: 2 **0 0** L9 true. L10 true. L11 not true.

See the <u>extended results file #3</u> for this variant. (As we see, the implication definition coincides here with the one used above to "discredit" $L_{10}$, but the negation definition is different.)

Under this variant the axioms $L_1$-$L_{10}$ are "true". As we know, under this variant, by MP, from "true" formulas only "true" formulas can be derived. The axiom $L_{11}$ is not "true" under this variant:

| B | ¬B | $B \vee \neg B$ |
|---|----|-----------------|
| 0 | 2  | 2               |
| 1 | 0  | 1               |
| 2 | 0  | 2               |

Hence, $L_{11}$ **cannot** be proved in $[L_1$-$L_{10}$, MP]. Q.E.D.

The results file #03 proves also the following

**Theorem 2.8.5 (**thanks to Pavels Mihailovs for a correction**).** The following (classically provable) formulas cannot be proved in the constructive propositional logic $[L_1$-$L_{10}$, MP]:

$$\neg\neg A \to A$$
$$(\neg B \to \neg A) \to (A \to B)$$
$$(\neg A \to B) \to (\neg B \to A)$$
$$(\neg\neg A \to \neg\neg B) \to (A \to B)$$
$$(A \to B) \to \neg A \vee B$$
$$((A \to B) \to B) \to A \vee B$$
$$((A \to B) \to A) \to A$$
$$\neg(A \wedge \neg B) \to (A \to B)$$
$$\neg(A \to B) \to A \wedge \neg B$$
$$A \vee (A \to B)$$

Indeed, all these formulas take non-"true" values under the "truth tables" from the proof of Theorem 2.8.4.

The following three formulas also cannot be proved in the constructive propositional logic, yet, unfortunately, the "truth tables" from our proof of Theorem 2.8.4 do not allow proving this:

$$\neg(A \wedge B) \to \neg A \vee \neg B$$
$$\neg\neg(A \vee B) \to \neg\neg A \vee \neg\neg B$$
$$(A \to B) \vee (B \to A)$$

Indeed, under the above "truth tables", these formulas always take "true" values (see results file #03). Another interesting conclusion: **add these three formulas as additional axioms to $[L_1$-$L_{10}$, MP] − and $L_{11}$ will remain still unprovable!**

Thus, we did not succeed in building a three-valued logic that would allow showing that the latter three formulas cannot be proved in the constructive propositional logic. Is it possible at all to build a multi-valued logic that would separate constructively provable propositional formulas from unprovable

ones? Kurt Gödel showed in 1932 that this is impossible: **none of the finitely-valued logics "matches" exactly the constructive propositional logic!**

**K. Gödel**. Zum intuitionistischen Aussagenkalkül, *Anzeiger Akademie der Wissenschaften Wien*, Math.-naturwiss. Klasse, 1932, Vol. 69, pp.65-66.

**Exercise 2.8.4**. a) (optional, for smart students) Verify somehow that the latter three formulas cannot be proved in the constructive propositional logic [$L_1$-$L_{10}$, MP]. Or, see Section 4.4.

b) Verify that any of the following formulas could be used – instead of $B \vee \neg B$ – as the axiom $L_{11}$ of the classical propositional logic: i) $(A \to B) \to \neg A \vee B$ , ii) $\neg\neg B \to B$, iii) $\neg(A \to B) \to A$ (Hint: since all these formulas are provable in [$L_1$-$L_{11}$, MP], it remains to prove $L_{11}$ in [$L_1$-$L_{10}$, MP] + (i), in [$L_1$-$L_{10}$, MP] + (ii), and in [$L_1$-$L_{10}$, MP] + (iii)).

c) Verify that with $\neg\neg B \to B$ instead of $L_{11}$ the "crazy" axiom $L_{10}$ becomes 100% derivable from the other axioms. Perhaps, this is why many textbooks prefer the combination $L_1$-$L_9$ + $\neg\neg B \to B$ as the axiom list for the classical propositional logic. But, then, we are forced to define the constructive propositional logic not as a subset of the classical one, but as the classical logic with the axiom $\neg\neg B \to B$ **replaced** by the "crazy" axiom $L_{10}$: $\neg B \to (B \to C)$!

## Axiom $L_{10}$ again...

Finally, let us check which of the main results of Sections 2.5 (constructive logic) and 2.6 (classical logic) depend on the "crazy" axiom $L_{10}$: $\neg A \to (A \to B)$. Let your program print out only those variants of "truth tables" that make "true" all the axioms $L_1$-$L_8$, and make: $L_9$ – "true", $L_{10}$ – not "true".

My program yields 6 such variants, see the results file #04. Surprisingly, in all these variants $L_{11}$ is "true" (thus, the results file #04 equals the results file #02). As the most productive appears

Implication variant #1:
2 2 2 0 2 2 0 1 2 L1-L8 true.
Variant #1. Negation: 2 2 1 L9 true. L10 not true. L11 true.


Constructively provable formulas:
Not true:  $(A \vee B) \to ((\neg A) \to B)$
Not true:  $((\neg A) \vee B) \to (A \to B)$
Not true: $((\neg\neg A) \to (\neg\neg B)) \to (\neg\neg(A \to B))$
Not true: $(\neg\neg A) \to ((\neg A) \to A)$
Not true:  $(A \vee (\neg A)) \to ((\neg\neg A) \to A)$
Not true: $\neg\neg((\neg\neg A) \to A)$


Classically provable formulas:
True:  $(\neg\neg(A \vee B)) \to ((\neg\neg A) \vee (\neg\neg B))$
True:  $(\neg(A \wedge B)) \to ((\neg A) \vee (\neg B))$
Not true: $(\neg\neg A) \to A$

Not true: ((¬B)→(¬A))→(A→B)
Not true: ((¬A)→B)→((¬B)→A)
Not true: ((¬¬A)→(¬¬B))→(A→B)
True:   $(A \to B) \to ((\neg A) \vee B)$
Not true:   $((A \to B) \to B) \to (A \vee B)$
Not true: ((A→B)→A)→A
Not true:   $(\neg(A \wedge (\neg B))) \to (A \to B)$
True: (A→B)→(((¬A)→B)→B)
Not true:   $(\neg(A \to B)) \to (A \wedge (\neg B))$
Not true:   $A \vee (A \to B)$
True:   $(A \to B) \vee (B \to A)$
Not true: (A→B)→(((¬A)→(¬B))→(B→A))

Thus, the following constructively provable formulas cannot be proved in the minimal logic [$L_1$-$L_9$, MP] (and even in [$L_1$-$L_9$, $L_{11}$, MP]), i.e. they cannot be proved without the "crazy" axiom $L_{10}$:

$$(A \vee B) \to (\neg A \to B)$$
$$\neg A \vee B \to (A \to B)$$
$$(\neg\neg A \to \neg\neg B) \to \neg\neg(A \to B)$$
$$\neg\neg A \to (\neg A \to A)$$
$$A \vee \neg A \to (\neg\neg A \to A)$$
$$\neg\neg(\neg\neg A \to A)$$

And the following classically provable formulas cannot be proved without the "crazy" axiom $L_{10}$ **(thanks to Pavels Mihailovs for a correction)**:

$$\neg\neg A \to A$$
$$(\neg B \to \neg A) \to (A \to B)$$
$$(\neg A \to B) \to (\neg B \to A)$$
$$(\neg\neg A \to \neg\neg B) \to (A \to B)$$
$$((A \to B) \to B) \to A \vee B$$
$$((A \to B) \to A) \to A$$
$$\neg(A \wedge \neg B) \to (A \to B)$$
$$\neg(A \to B) \to A \wedge \neg B$$
$$A \vee (A \to B)$$
$$(A \to B) \to ((\neg A \to \neg B) \to (B \to A))$$

**Exercise 2.8.5** (thanks to Stanislav Golubcov for the idea). But how about the remaining four (classically provable) formulas:

a)  $(A \to B) \to \neg A \vee B$ ,
b)  $\neg(A \wedge B) \to \neg A \vee \neg B$ ,
c)  $\neg\neg(A \vee B) \to \neg\neg A \vee \neg\neg B$ ,
$c_1$) (A→B)→((¬A→B)→B),

d) $(A \rightarrow B) \vee (B \rightarrow A)$  ?

Show that the formulas (a, b, c) **can be** proved without the "crazy" axiom $L_{10}$, i.e prove them in $[L_1\text{-}L_9, L_{11}, \text{MP}]$. (Hint: use Theorem 2.6.4 (a) $[L_1\text{-}L_8, \text{MP}]$: $A \vee \neg A \vdash (A \rightarrow B) \rightarrow \neg A \vee B.$ ). For smart students: how about the remaining formulas ($c_1$, d)?

**Using computers in mathematical proofs**

**Do you trust the above proofs?** Personally, I trust much more my ability to write (relatively) error-free computer programs than my ability to carry out error-free mathematical proofs. But how about you? Of course, you do not need trusting my (or your own) program generating the results files #00, #01, #02, #03 and #04. We used these files only to select the "truth table" variants allowing to prove our independence results. The critical points to be trusted are (see my implementation file) : a) the recursive program

<center>int MyFormula::ValueAt(int A, int B, int C)</center>

and b) the character string analyzer

<center>int MyFormula::Analyze(int *pOperation, AnsiString *pSubFormula1, AnsiString *pSubFormula2)</center>

You may wish to remove your worries by verifying directly that under all the 3 truth table variants used above: a) the axioms $L_1\text{-}L_8$ are true, and b) the axioms $L_9$, $L_{10}$, $L_{11}$ and other formulas are true or not true according to the goal of each particular proof. Before you have performed this 100%, you can feel **the flavor of using computers in mathematical proofs** (I trust this proof, I do not trust it, I trust this proof, I do not trust it, I trust this proof, I do not trust it, I trust this proof, I do not trust it, I trust this proof, I do not trust it, I trust this proof, I do not trust it, I trust this proof, I do not trust it, I trust this proof, I do not trust it, I trust this proof, I do not trust it, I trust this proof, I do not trust it, I trust this proof, I do not trust it, I trust this proof, I do not trust it,...)

Unfortunately, in more complicated cases the situation does not allow the above simple exit (i.e. manual verification of the solution found by a computer):

For pictures, see Four Colour Theorem in Wikipedia.

"The Four Colour Theorem was the first major theorem to be proved using a computer, having a proof that could not be verified directly by other mathematicians. Despite some worries about this initially, independent verification soon convinced everyone that the Four Colour Theorem had finally been proved. Details of the proof appeared in two articles in 1977. Recent work has led to improvements in the algorithm." (According to the article: The Four Colour Theorem in MacTutor History of Mathematics archive).

The proof of the Four Colour Theorem was completed in 1976 by Kenneth Appel and Wolfgang Haken.

"The best-known, and most debated, instance is the use of computer analysis by Kenneth Appel and Wolfgang Haken of the University of Illinois in their 1976 proof of the four-colour conjecture (that four colours suffice to colour in any map drawn upon a plane in such a way that countries which share a border are given different colours). First put forward in 1852, the conjecture had become perhaps the most famous unsolved problem in mathematics, resisting a multitude of efforts at proof for over a century. Appel and Haken's demonstration rested upon

computerized analysis, occupying 1,200 hours of computer time, of over 1,400 graphs. The analysis of even one of those graphs typically went beyond what an unaided human being could plausibly do: the ensemble of their demonstration certainly could not be checked in detail by human beings. In consequence, whether that demonstration constituted "proof" was deeply controversial..." (according to **Donald MacKenzie**. Computers and the Sociology of Mathematical Proof. In: *Trends in the History and Philosophy of Mathematics*, Odense: University of Southern Denmark, 2004, pp.67-86).

See also: Ken Appel on the 4CT proof, December 1998

The Four Color Theorem, November 13, 1995, by Robin Thomas.

A computer-checked proof of the Four Colour Theorem, 2004, by Georges Gonthier.

Two other famous computer assisted mathematical proofs:

- In 1989, by using Cray super-computers, Clement W. H. Lam finished his proof that finite projective plane of order 10 is impossible (for details see Projective plane in Wikipedia).

- In 1998, Thomas C. Hales finished his proof of Kepler conjecture about the densest arrangement of spheres in space (Johannes Kepler conjectured it in 1611, for details see Kepler conjecture in Wikipedia).

Visit the Mizar Project.

# 3. Predicate Logic

**Exercise 3.1.0**. Prove that the following derived rule (**Substitution Rule**) can be used in any of our predicate logics: if F(x/t) is an admissible substitution, then [$L_{12}$, MP, Gen]: F(x) ⊢ F(t). ("Can be used" means that any formula proved by using the Substitution Rule, can be proved without this rule as well.)

**Attention!** As you see, when assuming some formula F(x) as a hypothesis, one must have in mind that F(x) is "true for all values of the variable x". Else, avoid, in your proof, application of *Gen* and *Substitution Rule* to the variable x.

## 3.1. Proving Formulas Containing Quantifiers and Implication only

**Theorem 3.1.0.** [$L_1$, $L_2$, $L_{12}$, $L_{13}$, MP] ⊢ $\forall x B(x) \to \exists x B(x)$ . What does it mean? It prohibits "empty domains".

**Proof**. Indeed,

(1)    $\forall x B(x)$                          Hypothesis.

(2)    $\forall x B(x) \to B(x)$            Axiom $L_{12}$.

(3)   B(x)                                      By MP.

(4)    $B(x) \to \exists x B(x)$            Axiom $L_{13}$.

(5)    $\exists x B(x)$                          By MP.

Thus, by [$L_1$, $L_2$, MP] Deduction Theorem 2, there is a proof of [$L_1$, $L_2$, $L_{12}$, $L_{13}$, MP] ⊢ $\forall x B(x) \to \exists x B(x)$ . Q.E.D.

**Theorem 3.1.1.**

a) [$L_1$, $L_2$, $L_{12}$, $L_{14}$, MP, Gen] ⊢ $\forall x (B \to C) \to (\forall x B \to \forall x C)$ . What does it mean?

b) [$L_1$, $L_2$, $L_{12}$-$L_{15}$, MP, Gen] ⊢ $\forall$ x(B→C)→($\exists$ xB→$\exists$ xC). What does it mean?

**Proof**. Let us prove (a).

(1)  $\forall x(B{\to}C)$                      Hypothesis.

(2)  $\forall xB$                              Hypothesis.

(3)  $\forall x(B{\to}C){\to}(B{\to}C)$          Axiom $L_{12}$: $\forall xF(x){\to}F(x)$.

(4)  $B{\to}C$                                From (1) and (3), by MP.

(5)  $\forall xB{\to}B$                        Axiom $L_{12}$: $\forall xF(x){\to}F(x)$.

(6)  $B$                                      From (2) and (5), by MP.

(7)  $C$                                      From (4) and (6), by MP.

(8)  $\forall xC$                             From (7), by Gen.

In this proof, Gen is applied only to x, which is not a free variable in $\forall x(B{\to}C)$ and $\forall xB$. Thus, by $[L_1, L_2, L_{14}, MP, Gen]$ Deduction Theorem 2, there is a proof of $[L_1, L_2, L_{12}, L_{14}, MP, Gen] \vdash \forall x(B{\to}C) \to (\forall xB{\to}\forall xC)$.

Let us prove (b).

(1)  $\forall x(B{\to}C)$                      Hypothesis.

(2)  $\forall x(B{\to}C){\to}(B{\to}C)$          Axiom $L_{12}$: $\forall xF(x){\to}F(x)$.

(3)  $B{\to}C$                                From (1) and (2), by MP.

(4)  $C{\to}\exists xC$                        Axiom $L_{13}$: $F(x){\to} \exists xF(x)$.

(5)  $B{\to} \exists xC$                       From (3) and (4), by transitivity of implication $[L_1, L_2, MP]$.

(6)  $\forall x(B{\to} \exists xC)$            From (5), by Gen.

(7)  $\forall x(B{\to} \exists xC){\to}(\exists xB{\to} \exists xC)$    Axiom $L_{15}$: $\forall x(F(x){\to}G){\to}(\exists xF(x){\to}G)$ ($\exists xC$ does not contain x as a free variable).

(8)  $\exists xB{\to} \exists xC$              From (6) and (7), by MP.

In this proof, Gen is applied only to x, which is not a free variable in $\forall x(B{\to}C)$. Thus, by $[L_1, L_2, L_{14}, MP, Gen]$ Deduction Theorem 2, there is a

proof of $[L_1, L_2, L_{12}\text{-}L_{15}, MP, Gen] \vdash \forall x(B{\to}C) \to (\exists xB{\to} \exists xC)$.

Q.E.D.

**Theorem 3.1.2.**

a) $[L_1, L_2, L_5, L_{12}, L_{14}, MP, Gen] \vdash \forall x\forall yB(x, y){\leftrightarrow} \forall y\forall xB(x, y)$. What does it mean?

b) $[L_1, L_2, L_5, L_{13}, L_{15}, MP, Gen] \vdash \exists x\exists yB(x, y){\leftrightarrow} \exists y\exists xB(x, y)$. What does it mean?

c) $[L_1, L_2, L_{12}\text{-}L_{15}, MP, Gen] \vdash \exists x\forall yB(x, y){\to} \forall y\exists xB(x, y)$. What does it mean? The converse implication $\forall x\exists yB(x, y){\to} \exists y\forall xB(x, y)$ cannot be true. Explain, why.

**Proof**. To prove (a) and (c) do Exercise 3.1.1.

Let us prove (b).

| | | |
|---|---|---|
| (1) | $B(x, y){\to} \exists xB(x, y)$ | Axiom $L_{13}$ with $F(x) = B(x, y)$. |
| (2) | $\exists xB(x, y){\to} \exists y\exists xB(x, y)$ | Axiom $L_{13}$ with $F(y) = \exists xB(x, y)$. |
| (3) | $B(x, y){\to} \exists y\exists xB(x, y)$ | From (1) and (2), by transitivity of implication $[L_1, L_2, MP]$. |
| (4) | $F(x){\to}G \vdash \exists xF(x){\to}G$ | Exercise 1.4.3(a): $[L_{15}, MP, Gen]$, x not free in G. |
| (5) | $\exists yB(x, y){\to} \exists y \exists xB(x, y)$ | From (3), by (4), with $F(y) = B(x, y)$, $G = \exists x \exists yB(x, y)$. |
| (6) | $\exists x \exists yB(x, y){\to} \exists y \exists xB(x, y)$ | From (5), by (4), with $F(x) = \exists yB(x, y)$, $G = \exists x \exists yB(x, y)$. |

The proof of the converse implication $[L_1, L_2, L_{13}, L_{15}, MP, Gen] \vdash \exists y \exists xB(x, y){\to} \exists x \exists yB(x, y)$ is identical.

Now, by Axiom $L_5$ we obtain the equivalence (b). Q.E.D.

**Exercise 3.1.1.** Prove (a) and (c) of Theorem 3.1.2.

**Exercise 3.1.2.** Prove in the constructive logic,

$[L_1\text{-}L_{10}, L_{12}\text{-}L_{15}, MP, Gen] \vdash \exists x(B(x){\to}C(x)) \to (\forall xB(x){\to} \exists xC(x))$.

## 3.2. Formulas Containing Negations and a Single Quantifier

**Attention: non-constructive reasoning!** $\neg\forall\ xB \rightarrow \exists x\neg B$. This formula is accepted in the classical logic: if no x can possess the property B, then there is an x that does not possess B. It represents non-constructive reasoning in its ultimate form: let us assume, all x-s possess the property B, if we succeed in deriving a contradiction from this assumption, then – what? Is this a proof that there is a particular x that does not possess the property B? Does our proof contain a method allowing to build at least one such x? If not, do we have a "real" proof of $\exists x\neg B$?

How many formulas can be built of the formula B by using negations and a single quantifier?

$$\neg\neg\neg\neg\neg\neg\neg\neg\neg\forall x\neg\neg\neg\neg\neg\neg\neg\neg\neg B$$
$$\neg\neg\neg\neg\neg\neg\neg\neg\neg\exists x\neg\neg\neg\neg\neg\neg\neg\neg\neg B$$

Let us remind Theorem 2.4.5 [$L_1$-$L_9$, MP]: $\vdash \neg\neg\neg A \leftrightarrow \neg A$. I.e., any number of negations can be reduced to zero, one, or two, and thus we obtain $3*2*3 = 18$ formulas to be investigated. The following table represents the results of this investigation from

**A.Heyting.** On weakened quantification. *Journal of Symbolic Logic*, 1936, vol.11, pp.119-121 (see also Kleene [1952], Section 3.5).

**Table 3.2**

| I | III |
|---|---|
| $\forall xB$ | $\exists x\neg B$ |
| ------------------------------------------ | |
| $\neg\neg\forall xB$ | ---------------------------------------------- |
| ========================== | $\neg\neg\exists x\neg B$ |
| $\forall x\neg\neg B$ | $\neg\forall x\neg\neg B$ |
| $\neg\neg\forall x\neg\neg B$ | =========================== |
| $\neg\exists x\neg B$ | $\neg\forall xB$ |

| II | IV |
|---|---|
| ∃xB | |
| ------------------------------------------- | **IV** |
| ∃x¬¬B | ∀x¬B |
| ------------------------------------------- | ¬¬∀x¬B |
| ¬¬∃xB | ¬∃x¬¬B |
| ¬¬∃x¬¬B | ¬∃xB |
| ¬∀x¬B | |

**Legend**. a) In the classical logic, within each of the 4 groups all formulas are equivalent, for example, in group III: ¬∀xB↔∃x¬B. Of course, formulas of different groups cannot be equivalent (explain, why).

b) Two formulas within a group are constructively equivalent, if and only if they have no separating lines between them. For example, in group II: constructively, ¬∀x¬B↔¬¬∃xB, but not ¬∀x¬B↔∃xB (explain, why). All the formulas of the group IV are constructively equivalent.

c) If two formulas $F_1$, $F_2$ within a group ($F_1$ – above, $F_2$ – below) are separated by a **single line**, then: constructively, $F_1$→$F_2$, and ¬¬($F_2$→$F_1$), but not $F_2$→$F_1$. For example, in group II: constructively, ∃xB→¬∀x¬B, and ¬¬(¬∀x¬B→∃xB), but not ¬∀x¬B→∃xB (explain, why).

d) If two formulas $F_1$, $F_2$ within a group ($F_1$ – above, $F_2$ – below) are separated by a **double line**, then: constructively, $F_1$→$F_2$, but not $F_2$→$F_1$, and even not ¬¬($F_2$→$F_1$). For example, in group III: constructively, ∃x¬B→¬∀xB, but not ¬∀xB→∃x¬B, and even not ¬¬(¬∀xB→∃x¬B) (try to explain, why). Thus, the implication ¬∀xB→∃x¬B could be qualified as **super-non-constructive**.

Now, let us prove the implications necessary for the positive part of the above legend to be true.

**Note**. Proofs necessary for the negative part are not considered in the current version of this book.

**Group I**

I-1. Constructively, [$L_1$, $L_2$, $L_9$, MP]: ⊢ ∀xB→¬¬∀xB

Immediately, by Theorem 2.4.4 [$L_1$, $L_2$, $L_9$, MP]: ⊢A→¬¬A.

I-2. Constructively, [$L_1$-$L_9$, $L_{12}$, $L_{14}$, MP, Gen]: ⊢¬¬∀xB→∀x¬¬B

| | |
|---|---|
| (1) $\forall xB \to B$ | Axiom $L_{12}$: $\forall xF(x) \to F(x)$. |
| (2) $\neg\neg\forall xB \to \neg\neg B$ | From (1), by Theorem 2.4.7(a) [$L_1$-$L_9$, MP]: $\vdash (A \to B) \to (\neg\neg A \to \neg\neg B)$. |
| (3) $\forall x(\neg\neg\forall xB \to \neg\neg B)$ | From (2), by Gen. |
| (4) $\neg\neg\forall xB \to \forall x\neg\neg B$ | From (3), by Axiom $L_{14}$: $\forall x(G \to F(x)) \to (G \to \forall xF(x))$. |

I-3. Constructively, [$L_1$, $L_2$, $L_9$, MP]: $\vdash \forall x\neg\neg B \to \neg\neg\forall x\neg\neg B$

Immediately, by Theorem 2.4.4 [$L_1$, $L_2$, $L_9$, MP]: $\vdash A \to \neg\neg A$.

I-4. Constructively, [$L_1$, $L_2$, $L_9$, $L_{12}$, $L_{15}$, MP, Gen] $\vdash \neg\neg\forall x\neg\neg B \to \neg\exists x\neg B$

| | |
|---|---|
| (1) $\forall x\neg\neg B \to \neg\neg B$ | Axiom $L_{12}$: $\forall xF(x) \to F(x)$. |
| (2) $\neg\neg\neg B \to \neg\forall x\neg\neg B$ | From (1), by the Contraposition Law – Theorem 2.4.2. [$L_1$, $L_2$, $L_9$, MP]: $\vdash (A \to B) \to (\neg B \to \neg A)$. |
| (3) $\neg B \to \neg\neg\neg B$ | By Theorem 2.4.4 [$L_1$, $L_2$, $L_9$, MP]: $\vdash A \to \neg\neg A$. |
| (4) $\neg B \to \neg\forall x\neg\neg B$ | From (2) and (3), by transitivity of implication – Theorem 1.4.2 [$L_1$, $L_2$, MP]. |
| (5) $\forall x(\neg B \to \neg\forall x\neg\neg B)$ | From (4), by Gen. |
| (6) $\exists x\neg B \to \neg\forall x\neg\neg B$ | From (5), by Axiom $L_{15}$: $\forall x(F(x) \to G) \to (\exists xF(x) \to G)$. |
| (7) $\neg\neg\forall x\neg\neg B \to \neg\exists x\neg B$ | From (6), by the Contraposition Law [$L_1$, $L_2$, $L_9$, MP]. |

I-5. In the classical logic, [$L_1$-$L_{11}$, $L_{13}$, $L_{14}$, MP, Gen]: $\vdash \neg\exists x\neg B \to \forall xB$

| | |
|---|---|
| (1) $\neg B \to \exists x\neg B$ | Axiom $L_{13}$: $F(x) \to \exists xF(x)$. |
| (2) $\neg\exists x\neg B \to \neg\neg B$ | From (1), by the Contraposition Law [$L_1$, $L_2$, $L_9$, MP]. |

| | |
|---|---|
| (3) $\neg\neg B \rightarrow B$ | Classical logic, Theorem 2.6.1 [$L_1$-$L_{11}$, MP]: $\vdash \neg\neg A \rightarrow A$ |
| (4) $\neg\exists x \neg B \rightarrow B$ | From (2) and (3), by transitivity of implication [$L_1$, $L_2$, MP]. |
| (5) $\forall x(\neg\exists x \neg B \rightarrow B)$ | From (4), By Gen. |
| (6) $\neg\exists x \neg B \rightarrow \forall x B$ | From (5), by Axiom $L_{14}$: $\forall x(G \rightarrow F(x)) \rightarrow (G \rightarrow \forall x F(x))$. |

Thus, we have proved that in Group I, constructively, $F_1 \rightarrow F_2 \rightarrow F_3 \rightarrow F_4 \rightarrow F_5$, and, in the classical logic, $F_5 \rightarrow F_1$. I.e. we have proved that in Group I: a) in the classical logic, all the formulas are equivalent, and b) constructively, upper formulas imply lower formulas.

I-6. Constructively, [$L_1$, $L_2$, $L_9$, $L_{13}$, $L_{14}$, MP, Gen]: $\vdash \neg\exists x \neg B \rightarrow \forall x \neg\neg B$

| | |
|---|---|
| (1) $\neg B \rightarrow \exists x \neg B$ | Axiom $L_{13}$: $F(x) \rightarrow \exists x F(x)$. |
| (2) $\neg\exists x \neg B \rightarrow \neg\neg B$ | From (1), by the Contraposition Law [$L_1$, $L_2$, $L_9$, MP]. |
| (3) $\forall x(\neg\exists x \neg B \rightarrow \neg\neg B)$ | From (2), by Gen. |
| (4) $\neg\exists x \neg B \rightarrow \forall x \neg\neg B$ | From (3), by Axiom $L_{14}$: $\forall x(G \rightarrow F(x)) \rightarrow (G \rightarrow \forall x F(x))$. |

Thus, we have proved that in Group I, constructively, [$L_1$, $L_2$, $L_9$, $L_{12}$ -$L_{15}$, MP, Gen]: $F_3 \rightarrow F_4 \rightarrow F_5 \rightarrow F_3$, i.e. that formulas $F_3$, $F_4$, $F_5$ are constructively equivalent.

For Group I, it remains to prove

I-7. Constructively, [$L_1$-$L_{10}$, MP] $\vdash \neg\neg(\neg\neg\forall x B \rightarrow \forall x B)$

Immediately, by Theorem 2.5.2(d) [$L_1$-$L_{10}$, MP] $\vdash \neg\neg(\neg\neg A \rightarrow A)$.

**Group II**

II-1. Constructively, [$L_1$, $L_2$, $L_9$, $L_{12}$-$L_{15}$, MP, Gen] $\vdash \exists x B \rightarrow \exists x \neg\neg B$

| | |
|---|---|
| (1) $B \rightarrow \neg\neg B$ | By Theorem 2.4.4 [$L_1$, $L_2$, $L_9$, MP]: $\vdash A \rightarrow \neg\neg A$. |

(2)  $\forall x(B\rightarrow\neg\neg B)$     From (1), by Gen.

(3)  $\exists xB\rightarrow\exists x\neg\neg B$     From (2), by Theorem 3.1.1(b) [$L_1$, $L_2$, $L_{12}$-$L_{15}$, MP, Gen]

II-2. Constructively, [$L_1$-$L_9$, $L_{12}$-$L_{15}$, MP, Gen] $\vdash\exists x\neg\neg B\rightarrow\neg\neg\exists xB$

(1)  $B\rightarrow\exists xB$     Axiom $L_{13}$: $F(x)\rightarrow\exists xF(x)$.

(2)  $\neg\neg B\rightarrow\neg\neg\exists xB$     From (1), by Theorem 2.4.7(a) [$L_1$-$L_9$, MP]: $\vdash(A\rightarrow B)\rightarrow(\neg\neg A\rightarrow\neg\neg B)$.

(3)  $\forall x(\neg\neg B\rightarrow\neg\neg\exists xB)$     From (2), by Gen.

(4)  $\exists x\neg\neg B\rightarrow\neg\neg\exists xB$     From (3), by Theorem 3.1.1(b) [$L_1$, $L_2$, $L_{12}$-$L_{15}$, MP, Gen]

II-3. Constructively, [$L_1$-$L_9$, $L_{12}$-$L_{15}$, MP, Gen] $\vdash\neg\neg\exists xB\rightarrow\neg\neg\exists x\neg\neg B$

Immediately from II-1, by From (1), by Theorem 2.4.7(a) [$L_1$-$L_9$, MP]: $\vdash(A\rightarrow B)\rightarrow(\neg\neg A\rightarrow\neg\neg B)$.

II-4. Constructively, [$L_1$-$L_9$, $L_{12}$, $L_{15}$, MP, Gen] $\vdash\neg\neg\exists x\neg\neg B\rightarrow\neg\forall x\neg B$

(1)  $\forall x\neg B\rightarrow\neg B$     Axiom $L_{12}$: $\forall xF(x)\rightarrow F(x)$.

(2)  $\neg\neg B\rightarrow\neg\forall x\neg B$     From (1), by the Contraposition Law – Theorem 2.4.2. [$L_1$, $L_2$, $L_9$, MP]: $\vdash(A\rightarrow B)\rightarrow(\neg B\rightarrow\neg A)$.

(3)  $\forall x(\neg\neg B\rightarrow\neg\forall x\neg B)$     From (2), by Gen.

(4)  $\exists x\neg\neg B\rightarrow\neg\forall x\neg B$     From (3), by Axiom $L_{15}$: $\forall x(F(x)\rightarrow G)\rightarrow(\exists xF(x)\rightarrow G)$.

(5)  $\neg\neg\exists x\neg\neg B\rightarrow\neg\neg\neg\forall x\neg B$     From (4), by Theorem 2.4.7(a) [$L_1$-$L_9$, MP]: $\vdash(A\rightarrow B)\rightarrow(\neg\neg A\rightarrow\neg\neg B)$.

(6)  $\neg\neg\neg\forall x\neg B\rightarrow\neg\forall x\neg B$     Theorem 2.4.5 [$L_1$-$L_9$, MP]: $\vdash\neg\neg\neg A\leftrightarrow\neg A$

(7)  $\neg\neg\exists x\neg\neg B\rightarrow\neg\forall x\neg B$     From (5) and (6), by transitivity of implication [$L_1$, $L_2$, MP].

II-5. In the classical logic, $[L_1-L_{11}, L_{13}, L_{14}, MP, Gen]: \vdash \neg\forall x\neg B\rightarrow\exists xB$

| | |
|---|---|
| (1) $\neg\forall x\neg B\rightarrow\neg\neg\exists xB$ | II-6 $[L_1, L_2, L_9, L_{13}, L_{14}, MP, Gen]$, see below. |
| (2) $\neg\neg\exists xB\rightarrow\exists xB$ | Classical logic, Theorem 2.6.1 $[L_1-L_{11}, MP]: \vdash\neg\neg A \rightarrow A$ |
| (3) $\neg\forall x\neg B\rightarrow\exists xB$ | From (1) and (2), by transitivity of implication $[L_1, L_2, MP]$. |

Thus, we have proved that in Group II, constructively, $F_1\rightarrow F_2\rightarrow F_3\rightarrow F_4\rightarrow F_5$, and, in the classical logic, $F_5\rightarrow F_1$. I.e. we have proved that in Group II: a) in the classical logic, all the formulas are equivalent, and b) constructively, upper formulas imply lower formulas.

II-6. Constructively, $[L_1, L_2, L_9, L_{13}, L_{14}, MP, Gen] \vdash\neg\forall x\neg B\rightarrow\neg\neg\exists xB$

| | |
|---|---|
| (1) $B\rightarrow\exists xB$ | Axiom $L_{13}$: $F(x)\rightarrow\exists xF(x)$. |
| (2) $\neg\exists xB\rightarrow\neg B$ | From (1), by the Contraposition Law – Theorem 2.4.2. $[L_1, L_2, L_9, MP]: \vdash (A\rightarrow B)\rightarrow(\neg B\rightarrow\neg A)$. |
| (3) $\forall x(\neg\exists xB\rightarrow\neg B)$ | From (2), by Gen. |
| (4) $\neg\exists xB\rightarrow\forall x\neg B$ | From (3), by Axiom $L_{14}$: $\forall x(G\rightarrow F(x))\rightarrow(G\rightarrow\forall xF(x))$. |
| (5) $\neg\forall x\neg B\rightarrow\neg\neg\exists xB$ | From (4), by the Contraposition Law – Theorem 2.4.2. $[L_1, L_2, L_9, MP]: \vdash (A\rightarrow B)\rightarrow(\neg B\rightarrow\neg A)$. |

Thus, we have proved that in Group II, constructively, $[L_1-L_9, L_{12}-L_{15}, MP, Gen]$: $F_3\rightarrow F_4\rightarrow F_5\rightarrow F_3$, i.e. that formulas $F_3$, $F_4$, $F_5$ are constructively equivalent.

II-7. Constructively, $[L_1-L_{10}, MP]: \vdash\neg\neg(\neg\neg\exists xB\rightarrow\exists xB)$

Immediately, by Theorem 2.5.2 $[L_1-L_{10}, MP]: \vdash\neg\neg(\neg\neg A\rightarrow A)$.

Thus, constructively, $\neg\neg(F_3\rightarrow F_1)$, and $F_1\rightarrow F_2\rightarrow F_3\rightarrow F_4\rightarrow F_5\rightarrow F_3$. By Theorem 2.4.7(d), $[L_1-L_9, MP]$ $\neg\neg(A\rightarrow B)$, $\neg\neg(B\rightarrow C) \vdash\neg\neg(A\rightarrow C)$. Thus, in

fact, we have proved that in Group II, for all i, j, constructively, $\neg\neg(F_i \rightarrow F_j)$ (a kind of "weak equivalence").

**Group III**

III-1. Constructively, $[L_1, L_2, L_9, MP]: \vdash \exists x\neg B \rightarrow \neg\neg\exists x\neg B$

Immediately, by Theorem 2.4.4 $[L_1, L_2, L_9, MP]: \vdash A \rightarrow \neg\neg A$.

III-2. Constructively, $[L_1, L_2, L_9, L_{12}, L_{15}, MP, Gen]: \vdash \neg\neg\exists x\neg B \rightarrow \neg\forall x\neg\neg B$

| | | |
|---|---|---|
| (1) | $\forall x\neg\neg B \rightarrow \neg\neg\forall x\neg\neg B$ | I-3 $[L_1, L_2, L_9, MP]$, see above. |
| (2) | $\neg\neg\forall x\neg\neg B \rightarrow \exists x\neg B$ | I-4 $[L_1, L_2, L_9, L_{12}, L_{15}, MP, Gen]$, see above. |
| (3) | $\forall x\neg\neg B \rightarrow \exists x\neg B$ | From (1) and (2), by transitivity of implication $[L_1, L_2, MP]$. |
| (4) | $\neg\neg\exists x\neg B \rightarrow \neg\forall x\neg\neg B$ | From (3), by the Contraposition Law $[L_1, L_2, L_9, MP]$. |

III-3. Constructively, $[L_1\text{-}L_9, L_{12}, L_{14}, MP, Gen]: \vdash \neg\forall x\neg\neg B \rightarrow \neg\forall x B$

| | | |
|---|---|---|
| (1) | $\forall x B \rightarrow \neg\neg\forall x B$ | I-1 $[L_1, L_2, L_9, MP]$, see above. |
| (2) | $\neg\neg\forall x B \rightarrow \forall x\neg\neg B$ | I-2 $[L_1\text{-}L_9, L_{12}, L_{14}, MP, Gen]$ |
| (3) | $\forall x B \rightarrow \forall x\neg\neg B$ | From (1) and (2), by transitivity of implication $[L_1, L_2, MP]$. |
| (4) | $\neg\forall x\neg\neg B \rightarrow \neg\forall x B$ | From (3), by the Contraposition Law $[L_1, L_2, L_9, MP]$. |

III-4. In the classical logic, $[L_1\text{-}L_{11}, L_{13}, L_{14}, MP, Gen]: \vdash \neg\forall x B \rightarrow \exists x\neg B$

| | | |
|---|---|---|
| (1) | $\neg\exists x\neg B \rightarrow \forall x B$ | I-5: in the classical logic, $[L_1\text{-}L_{11}, L_{13}, L_{14}, MP, Gen]$ |
| (2) | $\neg\forall x B \rightarrow \neg\neg\exists x\neg B$ | From (1), by the Contraposition Law $[L_1, L_2, L_9, MP]$. |

(3) $\neg\neg\exists x\neg B\rightarrow\exists x\neg B$  Classical logic, Theorem 2.6.1 [$L_1$-$L_{11}$, MP]: $\vdash\neg\neg A\rightarrow A$

(4) $\neg\forall x B\rightarrow\exists x\neg B$  From (2) and (3), by transitivity of implication [$L_1$, $L_2$, MP].

Thus, we have proved that in Group III, constructively, $F_1\rightarrow F_2\rightarrow F_3\rightarrow F_4$, and, in the classical logic, $F_4\rightarrow F_1$. I.e. we have proved that in Group III: a) in the classical logic, all the formulas are equivalent, and b) constructively, upper formulas imply lower formulas.

III-4. Constructively, [$L_1$, $L_2$, $L_9$, $L_{13}$, $L_{14}$, MP, Gen]: $\vdash\neg\forall x\neg\neg B\rightarrow\neg\neg\exists x\neg B$

(1) $\neg\exists x\neg B\rightarrow\forall x\neg\neg B$  I-6 [$L_1$, $L_2$, $L_9$, $L_{13}$, $L_{14}$, MP, Gen]

(2) $\neg\forall x\neg\neg B\rightarrow\neg\neg\exists x\neg B$  From (1), by the Contraposition Law [$L_1$, $L_2$, $L_9$, MP].

Thus, we have proved that in Group III, constructively, $F_2\rightarrow F_3\rightarrow F_2$, i.e. that formulas $F_2$, $F_3$ are constructively equivalent.

III-5. Constructively, [$L_1$-$L_{10}$, MP]: $\vdash\neg\neg(\neg\neg\exists x\neg B\rightarrow\exists x\neg B)$

Immediately, by Theorem 2.5.2 [$L_1$-$L_{10}$, MP]: $\vdash\neg\neg(\neg\neg A\rightarrow A)$.

**Group IV**

IV-1. Constructively, [$L_1$, $L_2$, $L_9$, MP]: $\vdash\forall x\neg B\rightarrow\neg\neg\forall x\neg B$

Immediately, by Theorem 2.4.4 [$L_1$, $L_2$, $L_9$, MP]: $\vdash A\rightarrow\neg\neg A$.

IV-2. Constructively, [$L_1$-$L_9$, $L_{12}$-$L_{15}$, MP, Gen]: $\vdash\neg\neg\forall x\neg B\rightarrow\neg\exists x\neg\neg B$

(1) $\exists x\neg\neg B\rightarrow\neg\forall x\neg B$  From II-2, II-3, II-4 [$L_1$-$L_9$, $L_{12}$-$L_{15}$, MP, Gen], by transitivity of implication [$L_1$, $L_2$, MP].

(2) $\neg\neg\forall x\neg B\rightarrow\neg\exists x\neg\neg B$  From (1), by the Contraposition Law [$L_1$, $L_2$, $L_9$, MP].

IV-3. Constructively, [$L_1$, $L_2$, $L_9$, $L_{12}$-$L_{15}$, MP, Gen]: $\vdash\neg\exists x\neg\neg B\rightarrow\neg\exists x B$

(1) $\exists x B\rightarrow\exists x\neg\neg B$  II-1 [$L_1$, $L_2$, $L_9$, $L_{12}$-$L_{15}$, MP, Gen]

(2)  $\neg\exists x\neg\neg B\rightarrow\neg\exists xB$    From (1), by the Contraposition Law [$L_1$, $L_2$, $L_9$, MP].

IV-4. Constructively, [$L_1$, $L_2$, $L_9$, $L_{13}$, $L_{14}$, MP, Gen]: $\vdash\neg\exists xB\rightarrow\forall x\neg B$

(1)  $B\rightarrow\exists xB$    Axiom $L_{13}$: $F(x)\rightarrow\exists xF(x)$.

(2)  $\neg\exists xB\rightarrow\neg B$    From (1), by the Contraposition Law [$L_1$, $L_2$, $L_9$, MP].

(3)  $\forall x(\neg\exists xB\rightarrow\neg B)$    From (2), by Gen.

(4)  $\neg\exists xB\rightarrow\forall x\neg B$    From (3), by Axiom $L_{14}$: $\forall x(G\rightarrow F(x))\rightarrow(G\rightarrow\forall xF(x))$.

Thus, we have proved that in Group IV all the formulas are constructively equivalent.

And thus, we have proved the positive part of the legend of Table 3.2. The negative part of the legend asserts that the following (classically provable) formulas cannot be proved constructively:

(1)  $\neg\neg\forall xB\rightarrow\forall xB$    See Group I. Simply, an instance of (the non-constructive) $\neg\neg A\rightarrow A$.

(2)  $\forall x\neg\neg B\rightarrow\neg\neg\forall xB$    See Group I. **Super-non-constructive**: even $\neg\neg(2)$ is non-constructive!

(3)  $\neg\neg(\forall x\neg\neg B\rightarrow\neg\neg\forall xB)$    $\neg\neg(2)$. See Group I.

(4)  $\exists x\neg\neg B\rightarrow\exists xB$    See Group II. Nearly, an instance of (the non-constructive) $\neg\neg A\rightarrow A$.

(5)  $\neg\neg\exists xB\rightarrow\exists x\neg\neg B$    See Group II. Stronger than simply non-constructivity of $\neg\neg A\rightarrow A$?

(6)  $\neg\neg\exists x\neg B\rightarrow\exists x\neg B$    See Group III. Simply, an instance of (the non-constructive) $\neg\neg A\rightarrow A$.

(7)  $\neg\forall xB\rightarrow\neg\forall x\neg\neg B$    See Group III. **Super-non-constructive**: even $\neg\neg(7)$ is non-constructive!

(8)  $\neg\neg(\neg\forall xB\rightarrow\neg\forall x\neg\neg B)$    $\neg\neg(7)$. See Group III.

Still, the most striking (classically provable) non-constructive quantifier implications correspond to existence proofs via *reductio ad absurdum*:

| | | |
|---|---|---|
| (8) | $\neg\forall x\neg B\to\exists xB$ | $\neg\neg$(8) is constructively provable, but (8) is not, see Group II. If we know how to derive a contradiction from $\forall x\neg B$, then may be, we do not know how to find a particular x such that B. |
| (9) | $\neg\forall x\neg B\to\neg\neg\exists x\neg\neg B$ | (9) is weaker than (8), but still non-constructive, see Group II. If we know how to derive a contradiction from $\forall x\neg B$, then may be, we do not know how to derive a contradiction from $\neg\exists x\neg\neg B$. |
| (10) | $\neg\forall xB\to\exists x\neg B$ | $\exists$ven $\neg\neg$(10) is non-constructive, see Group III. If we know how to derive a contradiction from $\forall xB$, then may be, we do not know how to find a particular x such that $\neg B$. |
| (11) | $\neg\forall xB\to\neg\neg\exists x\neg B$ | (11) is weaker than (10), but still super-non-constructive (i.e. even $\neg\neg$(11) is non-constructive), see Group III. If we know how to derive a contradiction from $\forall xB$, then may be, we do not know how to derive a contradiction from $\neg\exists x\neg B$. |

## 3.3. Proving Formulas Containing Conjunction and Disjunction

**Theorem 3.3.1.**

a) [$L_1$-$L_5$, $L_{12}$, $L_{14}$, MP, Gen]: $\vdash \forall x(B\wedge C)\leftrightarrow\forall xB\wedge\forall xC$ .

b) [$L_1$, $L_2$, $L_6$-$L_8$, $L_{14}$, MP, Gen]: $\vdash \forall xB\vee\forall xC\to\forall x(B\vee C)$ . The converse formula $\forall x(B\vee C)\to\forall xB\vee\forall xC$ cannot be true. Explain, why.

**Proof**. Before proving (a), do Exercise 3.3.1.

Since, in your first proof, Gen has been applied only to x, which is not a free variable in $\forall x(B\wedge C)$ , then, by Deduction theorem 2 [$L_1$, $L_2$, $L_{14}$, MP, Gen] we obtain that

$[L_1\text{-} L_5, L_{12}, L_{14}, \text{MP, Gen}]: \vdash \forall x(B \wedge C) \rightarrow \forall xB \wedge \forall xC$ .

Similarly, in your second proof, Gen has been applied only to x, which is not a free variable in $\forall xB \wedge \forall xC$ , then, by Deduction theorem 2 $[L_1, L_2, L_{14},$ MP, Gen] we obtain that

$[L_1\text{-} L_5, L_{12}, L_{14}, \text{MP, Gen}]: \vdash \forall xB \wedge \forall xC \rightarrow \forall x(B \wedge C)$ .

Now, by Theorem 2.2.1(a) $[L_5]$: A, B $\vdash A \wedge B$ , we obtain the equivalence (a) of Theorem 3.3.1.

To prove (b) do Exercise 3.3.2. Q.E.D.

**Exercise 3.3.1.** Prove $[L_3\text{-}L_5, L_{12}, \text{MP, Gen}]$: $\forall x(B \wedge C) \vdash \forall xB \wedge \forall xC$ and $[L_3\text{-}L_5, L_{12}, \text{MP, Gen}]$: $\forall xB \wedge \forall xC \vdash \forall x(B \wedge C)$ .

**Exercise 3.3.2.** Use Theorem 2.3.1 to prove (b) of Theorem 3.3.1.

**Theorem 3.3.2.**

a) $[L_1\text{-}L_8, L_{12}\text{-}L_{15}, \text{MP, Gen}]: \vdash \exists x(B \vee C) \leftrightarrow \exists xB \vee \exists xC$ .

b) $[L_1\text{-}L_5, L_{13}\text{-}L_{15}, \text{MP, Gen}]: \vdash \exists x(B \wedge C) \rightarrow \exists xB \wedge \exists xC$ . The converse implication $\exists xB \wedge \exists xC \rightarrow \exists x(B \wedge C)$ cannot be true. Explain, why.

**Proof**. Do Exercise 3.3.3.

**Exercise 3.3.3.** a) Prove (a→) of Theorem 3.3.2. (Hint: start with $L_{13}$, $L_6$, $L_7$, Gen, and finish by applying $L_{15}$.)

b) Prove (a←) of Theorem 3.3.2. (Hint: apply Theorem 2.3.1 and Theorem 3.3.1)

c) Prove (b) of Theorem 3.3.2. (Hint: first, assume $B \wedge C$ , derive $\exists xB \wedge \exists xC$ , and apply Deduction Theorem 1).

## 3.4. Replacement Theorems

An example: we know that $\log xy = \log x + \log y$ . Hence,

$$\log 2^a 3^b = \log 2^a + \log 3^b$$ .

The latter formula represents an instance of the former one. We know also that $\log x^y = y \cdot \log x$ , hence:

$$\log 2^a 3^b = \log 2^a + \log 3^b = a \cdot \log 2 + b \cdot \log 3$$ .

Here we applied the formula $\log x^y = y \cdot \log x$ to *sub-formulas*.

Another example: we know the following theorem of the classical logic: $\vdash (A \to B) \leftrightarrow \neg A \vee B$ . Hence,

$$(X \to Y) \to Z \leftrightarrow \neg (X \to Y) \vee Z \ .$$

But the next step, obtaining the formula $\neg(\neg X \vee Y) \vee Z$ means already that the above theorem is applied to a *sub-formula*.

We know also that $\neg(A \vee B) \leftrightarrow \neg A \wedge \neg B$ , hence, we can continue, obtaining $(\neg \neg X \wedge \neg Y) \vee Z$ , and $(X \wedge \neg Y) \vee Z$ (since $\vdash \neg \neg A \leftrightarrow A$). Again, we have applied an equivalence theorem to a *sub-formula*. We would like to conclude that the final result is equivalent to the initial formula:

$$(X \to Y) \to Z \leftrightarrow (X \wedge \neg Y) \vee Z \ .$$

Until now, in our logic, we could not use this very natural kind of mathematical argument.

In this section we will prove meta-theorems that will allow replacing sub-formulas by equivalent formulas. For example, if have proved the formula $\exists x B \to D$, and we know that $\vdash B \leftrightarrow C$, then we can replace B by C, obtaining the formula $\exists x C \to D$. These theorems will make the above treatment of the formula (X→Y)→Z completely legal.

We will prove also that the meaning of a formula does not depend on the names of bound variables used in it. For example,

$$\vdash (\exists x B(x) \to C) \leftrightarrow (\exists y B(y) \to C).$$

**Note**. To prove all these replacement theorems we will need only the **minimal logic** [$L_1$-$L_9$, $L_{12}$-$L_{15}$, MP, Gen].

**Sub-formulas and Occurrences**

Intuitively, B is a sub-formula of C, if B is a formula, and B is a part (substring) of C. But note that a sub-formula may appear in the same formula more than once, as, for example, in the following instance of the axiom $L_1$:

**∃xB(x)→(∃xC(x)→∃xB(x))**. Thus, it would be more correctly to speak about **occurrences** of sub-formulas. In the above example, there are two occurrences of the formula ∃xB(x).

The formal definition is as follows:

a) o(B) is an occurrence in B in B.

b) If o(B) is an occurrence of B in C, then o(B) is an occurrence of B in ¬C, $C \wedge D$, $D \wedge C$, $C \vee D$, $D \vee C$ , C→D, and D→C.

b) If o(B) is an occurrence of B in C, then o(B) is an occurrence of B in ∃xC, and ∀xC.

We can define also the notion of **propositional** occurrences:

a) o(B) is a propositional occurrence in B in B.

b) If o(B) is a propositional occurrence of B in C, then o(B) is a propositional occurrence of B in ¬C, $C \wedge D$, $D \wedge C$, $C \vee D$, $D \vee C$, C→D, and D→C.

Intuitively, o(B) is a propositional occurrence of B in C, if, in C, no quantifiers stand over o(B).

**Replacement Lemma 1.** In the minimal logic, $[L_1\text{-}L_9, MP]$:

(a)  A↔B $\vdash$ (A→C)↔(B→C)          $[L_1\text{-}L_5, MP]$

(b)  A↔B $\vdash$ (C→A)↔(C→B)          $[L_1\text{-}L_5, MP]$

(c)  A↔B $\vdash$ $A \wedge C \leftrightarrow B \wedge C$          $[L_1\text{-}L_5, MP]$

(d)  A↔B $\vdash$ $C \wedge A \leftrightarrow C \wedge B$          $[L_1\text{-}L_5, MP]$

(e)  A↔B $\vdash$ $A \vee C \leftrightarrow B \vee C$          $[L_1\text{-}L_8, MP]$

(f)  A↔B $\vdash$ $C \vee A \leftrightarrow C \vee B$          $[L_1\text{-}L_8, MP]$

(g)  A↔B $\vdash$ ¬A↔¬B          $[L_1\text{-}L_9, MP]$

**Proof.** To prove (a) we will first prove that $[L_1, L_2, L_4, MP]$: A↔B $\vdash$ (A→C)→(B→C).

(1)  $(A \to B) \wedge (B \to A)$          A↔B – hypothesis.

(2)  A→C          Hypothesis.

(3)  B→A          From (1), by Axiom $L_4$.

(4)  B→C          From (3) and (2), by transitivity of implication $[L_1, L_2, MP]$.

Thus, by $[L_1, L_2, MP]$ Deduction Theorem 1, $[L_1, L_2, L_4, MP]$: A↔B $\vdash$ (A→C)→(B→C).

In a similar way, we can prove that

$$[L_1, L_2, L_3, MP]: A \leftrightarrow B \vdash (B \to C) \to (A \to C).$$

Now, by Theorem 2.2.1(a), we obtain (a).

To prove other cases do Exercises 3.4.1, 3.4.2.

Q.E.D.

**Exercise 3.4.1.** Prove (b, c, d ) of Replacement Lemma 1.

**Exercise 3.4.2.** Prove (e, f, g) of Replacement Lemma 1.

**Replacement Theorem 1.** Let us consider three formulas: B, B', C, where B is a sub-formula of C, and o(B) is a **propositional** occurrence of B in C (i.e. no quantifiers stand over o(B)). Let us denote by C' the formula obtained from C by replacing o(B) by B' . Then, in the minimal logic,

$$[L_1\text{-}L_9,\ MP]:\ B\leftrightarrow B'\vdash C\leftrightarrow C'.$$

**Proof**. Induction by the "depth" of the propositional occurrence o(B).

**Induction base:** depth = 0. Then C is B, and C' is B'. The conclusion is obvious.

**Induction step.** If C is not B, then one of the following holds:

a) C is F→G, and o(B) is in F.

b) C is F→G, and o(B) is in G.

c) C is $F\wedge G$ , and o(B) is in F.

d) C is $F\wedge G$ , and o(B) is in G.

e) C is $F\vee G$ , and o(B) is in F.

f) C is $F\vee G$ , and o(B) is in G.

g) C is ¬F, and o(B) is in F.

Case (a). By induction assumption, $[L_1\text{-}L_9,\ MP]:\ B\leftrightarrow B'\ \vdash\ F\leftrightarrow F'$. By Replacement Lemma 1(a), $[L_1\text{-}L_9,\ MP]:\ F\leftrightarrow F'\vdash(F\rightarrow G)\leftrightarrow(F'\rightarrow G)$. Thus,

$$[L_1\text{-}L_9,\ MP]:\ B\leftrightarrow B'\vdash C\leftrightarrow C'.$$

To complete the proof do Exercise 3.4.3.

Q.E.D.

**Exercise 3.4.3.** Repeat the above argument for the remaining cases (b, c, d, e, f, g).

Now, we can use the replacement argument mentioned at the beginning of this section – at least, for propositional occurrences of equivalent sub-formulas.

**Replacement Lemma 2.** In the minimal logic, $[L_1\text{-}L_9,\ L_{12}\text{-}L_{15},\ MP,\ Gen]$:

(a)  $B \leftrightarrow C \vdash \forall xB \leftrightarrow \forall xC$          $[L_1\text{-}L_5, L_{12}, L_{14}, MP, Gen]$

(b)  $B \leftrightarrow C \vdash \exists xB \leftrightarrow \exists xC$          $[L_1\text{-}L_5, L_{12}\text{-}L_{15}, MP, Gen]$

**Exercise 3.4.4**. Prove Replacement Lemma 2.

**Replacement Theorem 2.** Let us consider three formulas: B, B', C, where B is a sub-formula of C, and o(B) is **any** occurrence of B in C. Let us denote by C' the formula obtained from C by replacing o(B) by B' . Then, in the minimal logic,

$$[L_1\text{-}L_9, L_{12}\text{-}L_{15}, MP, Gen]: B \leftrightarrow B' \vdash C \leftrightarrow C'.$$

**Proof**. Induction by the "depth" of the occurrence o(B).

**Induction base:** depth = 0. Then C is B, and C' is B'. The conclusion is obvious.

**Induction step.** If C is not B, then one of the following holds:

a)-g) – as in the proof of Replacement Theorem 1.

h) C is $\forall xF$, and o(B) is in F.

i) C is $\exists xF$, and o(B) is in F.

**Case (h).** By induction assumption, $[L_1\text{-}L_9, L_{12}\text{-}L_{15}, MP, Gen]: B \leftrightarrow B' \vdash F \leftrightarrow F'$. By Replacement Lemma 2(a), $[L_1\text{-}L_9, L_{12}\text{-}L_{15}, MP, Gen]: F \leftrightarrow F' \vdash \forall xF \leftrightarrow \forall xF'$. Thus, $[L_1\text{-}L_9, L_{12}\text{-}L_{15}, MP, Gen]: B \leftrightarrow B' \vdash C \leftrightarrow C'$.

**Case (i).** By induction assumption, $[L_1\text{-}L_9, L_{12}\text{-}L_{15}, MP, Gen]: B \leftrightarrow B' \vdash F \leftrightarrow F'$. By Replacement Lemma 2(b), $[L_1\text{-}L_9, L_{12}\text{-}L_{15}, MP, Gen]: F \leftrightarrow F' \vdash \exists xF \leftrightarrow \exists xF'$. Thus, $[L_1\text{-}L_9, L_{12}\text{-}L_{15}, MP, Gen]: B \leftrightarrow B' \vdash C \leftrightarrow C'$.

Q.E.D.

**Now (only now!) , we may use in our proofs the replacement argument mentioned at the beginning of this section. And, for any equivalent sub-formulas!**

Finally, let us prove that the meaning of a formula does not depend on the names of bound variables used in it. Intuitively, it "must be so", but now we can prove this intuition as a meta-theorem.

**Replacement Lemma 3.** If the formula B does not contain the variable y, then (in the minimal logic):

a) $[L_5, L_{12}, L_{14}, MP, Gen]: \vdash \forall xB(x) \leftrightarrow \forall yB(y)$

b) $[L_5, L_{13}, L_{15}, MP, Gen]$: $\vdash \exists xB(x) \leftrightarrow \exists yB(y)$.

**Proof**. First, let us prove $[L_{12}, L_{14}, MP, Gen]$: $\vdash \forall xB(x) \rightarrow \forall yB(y)$.

| | | |
|---|---|---|
| (1) | $\vdash \forall xB(x) \rightarrow B(y)$ | Axiom $L_{12}$: $\forall xF(x) \rightarrow F(t)$. $B(x)$ does not contain y, hence, $B(x/y)$ is an admissible substitution. |
| (2) | $\vdash \forall y(\forall xB(x) \rightarrow B(y))$ | By Gen. |
| (3) | $\vdash \forall y(\forall xB(x) \rightarrow B(y)) \rightarrow (\forall xB(x) \rightarrow \forall yB(y))$ | Axiom $L_{14}$: $\forall x(G \rightarrow F(x)) \rightarrow (G \rightarrow \forall xF(x))$. $\forall xB(x)$ does not contain y. |
| (4) | $\vdash \forall xB(x) \rightarrow \forall yB(y)$ | By MP. |

Now, let us prove $[L_{12}, L_{14}, MP, Gen(x)]$: $\vdash \forall yB(y) \rightarrow \forall xB(x)$.

| | | |
|---|---|---|
| (1) | $\vdash \forall yB(y) \rightarrow B(x)$ | Axiom $L_{12}$: $\forall xF(x) \rightarrow F(t)$. $B(x)$ does not contain y, hence, $B(y)$ contains only free occurrences of y, i.e. $B(y/x)$ is an admissible substitution. |
| (2) | $\vdash \forall x(\forall yB(y) \rightarrow B(x))$ | By Gen. |
| (3) | $\vdash \forall x(\forall yB(y) \rightarrow B(x)) \rightarrow (\forall yB(y) \rightarrow \forall xB(x))$ | Axiom $L_{14}$: $\forall x(G \rightarrow F(x)) \rightarrow (G \rightarrow \forall xF(x))$. $\forall yB(y)$ does not contain x as a free variable. |
| (4) | $\vdash \forall yB(y) \rightarrow \forall xB(x)$ | By MP. |

Now, by Theorem 2.2.1(a), we obtain (a).

To prove (b), first, let us prove $[L_{13}, L_{15}, MP, Gen(y)]$: $\vdash \exists yB(y) \rightarrow \exists xB(x)$.

| | | |
|---|---|---|
| (1) | $\vdash B(y) \rightarrow \exists xB(x)$ | Axiom $L_{13}$: $F(t) \rightarrow \exists xF(t)$. $B(x)$ does not contain y, hence, $B(x/y)$ is an admissible substitution. |
| (2) | $\vdash \forall y(B(y) \rightarrow \exists xB(x))$ | By Gen. |

| (3) | ⊢∀y(B(y)→∃xB(x))→(∃yB(y)→∃xB(x)) | Axiom $L_{15}$: <br> $\forall x(F(x)\to G)\to(\exists xF(x)\to G)$. $\exists xB(x)$ does not contain y. |
| (4) | ⊢∃yB(y)→∃xB(x) | By MP. |

Now, let us prove [$L_{13}$, $L_{15}$, MP, Gen(x)]: ⊢∃xB(x)→∃yB(y).

| (1) | ⊢B(x)→∃yB(y) | Axiom $L_{13}$: F(t)→∃xF(x). B(x) does not contain y, hence, B(y) contains only free occurrences of y, i.e. B(y/x) is an admissible substitution. |
| (2) | ⊢ ∀x(B(x)→∃yB(y)) | By Gen. |
| (3) | ⊢∀x(B(x)→∃yB(y))→(∃xB(x)→∃yB(y)) | Axiom $L_{15}$: <br> $\forall x(F(x)\to G)\to(\exists xF(x)\to G)$. $\exists yB(y)$ does not contain x as a free variable. |
| (4) | ⊢∃xB(x)→∃yB(y) | By MP. |

Now, by Theorem 2.2.1(a), we obtain (b).

Q.E.D.

**Replacement Theorem 3.** Let *y* be a variable that does not occur in a formula F, containing an occurrence of a quantifier ∀x (or ∃x). Let us replace by *y* all occurrences of the variable *x* bound by this particular quantifier occurrence. Let us denote the resulting formula by F'. Then, in the minimal logic,

$$[L_1\text{-}L_9, L_{12}\text{-}L_{15}, \text{MP, Gen}]: \vdash F\leftrightarrow F'.$$

**Proof.** Thus, the formula F contains a sub-formula ∀xB(x) (or ∃xB(x)), and we wish to replace it by ∀y(B(y)) (or ∃yB(y)), where y does not occur in F. By Replacement Lemma 3, in the minimal logic, ⊢ ∀xB(x)↔∀yB(y), and ⊢ ∃xB(x)↔∃yB(y). Hence, by Replacement Lemma 2, in the minimal logic, ⊢ F↔F'. Q.E.D.

Now let us repeat our example. We know that (in the classical logic): ⊢ $(A\to B)\leftrightarrow\neg A\vee B$ . Hence, the formula (X→Y)→Z **is** equivalent to $\neg(X\to Y)\vee Z$ , and to $\neg(\neg X\vee Y)\vee Z$ . We know also that $\neg(A\vee B)\leftrightarrow\neg A\wedge\neg B$ , hence, we can continue: (X→Y)→Z **is** equivalent to $(\neg\neg X\wedge\neg Y)\vee Z$ , and to $(X\wedge\neg Y)\vee Z$ (since ⊢¬¬A↔A).

Now, in our logic, we can use this very natural kind of mathematical argument.

## 3.5. Constructive Embedding

Glivenko's Theorem (see Section 2.7) provides a simple "constructive embedding" for the classical propositional logic: any classically provable formula can be "proved" in the constructive logic, if you put two negations before it. This theorem does not hold for the predicate logic. For example (see Section 3.2),

II-5. In the classical logic, $[L_1\text{-}L_{11}, L_{13}, L_{14}, MP, Gen]$: $\vdash \neg\forall x \neg B \rightarrow \exists x B$.

The double negation of this formula, i.e. the formula $\neg\neg(\neg\forall x\neg B \rightarrow \exists x B)$ cannot be proved in the constructive predicate logic. Thus, instead of the simple operation $\neg\neg F$, we must search for a more complicated embedding operation.

However,

**Exercise 3.5.1** (optional, for smart students). Verify that a formula F is provable in the classical predicate logic, if and only if $\neg\neg F$ is provable in the constructive predicate logic plus the following axiom schema: $\forall x\neg\neg B \rightarrow \neg\neg\forall x B$ (the so-called *Double Negation Shift s*chema, see Intuitionistic Logic by **Joan Moschovakis** in Stanford Encyclopedia of Philosophy.

The first embedding operation was introduced by Andrey Nikolaevich Kolmogorov (1903-1987) in

**A.N.Kolmogorov**. On the principle tertium non datur. *Matem. sbornik*, 1925, vol.32, pp.646-667 (in Russian).

A quote from A Short Biography of A.N. Kolmogorov by Paul M.B. Vitanyi follows:

"K. got interested in mathematical logic, and in 1925 published a paper in Mathematicheskii Sbornik on the law of the excluded middle, which has been a continuous source for later work in mathematical logic. This was the first Soviet publication on mathematical logic containing (very substantial) new results, and the first systematic research in the world on intuitionistic logic. K. anticipated to a large extent A. Heyting 's formalization of intuitionistic reasoning, and made a more definite correlation between classical and intuitionistic mathematics. K. defined an operation for `embedding' one logical theory in another. Using this – historically the first such operation, now called the `Kolmogorov operation' – to embed classical logic in intuitionistic logic, he proved that application of the law of the excluded middle in itself cannot lead to a contradiction."

See also Kolmogorov Centennial.

We will investigate the following version of an embedding operation: to obtain O(F), in a formula F, put two negations before: a) every atomic formula, b) every disjunction, c) every existential quantifier. More precisely, let us define the following embedding operation O (you may wish to compare it with some other versions possessing similar properties):

| Operation O Detlovs [1964] | Operation K Kolmogorov [1925] | Operation O' Gödel [1933], see Kleene [1952] | Operation O$^o$ Gentzen [1936], see Kleene [1952] |
|---|---|---|---|
| If F is an atomic formula, then O(F) is ¬¬F. | K(F) is ¬¬F. | O'(F) is F. | O$^o$(F) is F. |
| O(F→G) is O(F)→O(G). | ¬¬(K(F)→K(G)) | $\neg(O'(F)\wedge\neg O'(G))$ | O$^o$(F)→O$^o$(G) |
| $O(F\wedge G)$ is $O(F)\wedge O(G)$ . | $\neg\neg(K(F)\wedge K(G))$ | $O'(F)\wedge O'(G)$ | $O^o(F)\wedge O^o(G)$ |
| $O(F\vee G)$ is $\neg\neg(O(F)\vee O(G))$ | $\neg\neg(K(F)\vee K(G))$ | $\neg(\neg O'(F)\wedge\neg O'(G))$ | $\neg(\neg Oo(F)\wedge\neg Oo(G))$ |
| O(¬F) is ¬O(F). | ¬¬¬K(F), or ¬K(F)[*] | ¬O'(F) | ¬O$^o$(F) |
| O(∀xF) is ∀xO(F). | ¬¬∀xK(F) | ∀xO'(F) | ∀xO$^o$(F) |
| O(∃xF) is ¬¬∃xO(F). | ¬¬∃xK(F) | ¬∀x¬O'(F) | ¬∀x¬O$^o$(F) |

([*]) By Theorem 2.4.5, [L$_1$-L$_9$, MP]: ⊢¬¬¬K(F)↔¬K(F).

For example, let us take the above formula ¬∀x¬B→∃xB. If B is an atomic formula, then

O(¬∀x¬B→∃xB) is ¬∀x¬¬¬B→¬¬∃x¬¬B, i.e. ¬∀x¬B→¬¬∃x¬¬B

The latter formula is constructively provable (see Section 3.2, Group II).

**Lemma 3.5.1**. For any formula F, in the classical logic, ⊢F↔O(F).

**Proof.** By induction. Let us remind Theorem 2.6.1: [L$_1$-L$_{11}$, MP] ⊢¬¬A ↔ A.

**1. Induction base**: F is an atomic formula. Then O(F) is ¬¬F. By Theorem 2.6.1, [L$_1$-L$_{11}$, MP] ⊢¬¬F↔F, hence, in the classical logic, ⊢O(F)↔F.

**2. Induction step.**

**Case 2a:** F is B∨C. Then O(F) is ¬¬(O(B)∨O(C)).

| | | |
|---|---|---|
| (1) O(B)↔B | | Induction assumption. |
| (2) O(C)↔C | | Induction assumption. |
| (3) $B\vee C\leftrightarrow O(B)\vee C$ | | From (1), by Replacement Theorem 1. |

| | |
|---|---|
| (4) $O(B)\lor C\leftrightarrow O(B)\lor O(C)$ | From (2), by Replacement Theorem 1. |
| (5) $O(B)\lor O(C)\leftrightarrow\neg\neg(O(B)\lor O(C))$ | Theorem 2.6.1: $[L_1\text{-}L_{11}, MP]\vdash$ $\neg\neg A\leftrightarrow A$. |
| (6) $B\lor C\leftrightarrow\neg\neg(O(B)\lor O(C))$ , i.e. F↔O(F) | By transitivity of implication. |

**Case 2b:** F is ∃xB. Then O(F) is $\neg\neg\exists xO(B)$.

| | |
|---|---|
| (1) O(B)↔B | Induction assumption. |
| (2) ∃xB↔∃xO(B) | From (1), by Replacement Theorem 2. |
| (3) $\exists xO(B)\leftrightarrow\neg\neg\exists xO(B)$ | Theorem 2.6.1: $[L_1\text{-}L_{11}, MP]\vdash\neg\neg A\leftrightarrow A$. |
| (4) $\exists xB\leftrightarrow\neg\neg\exists xO(B)$, i.e. F↔O(F) | By transitivity of implication. |

**Case 2c:** F is B→C.

**Case 2d:** F is B&C.

**Case 2e:** F is ¬B.

**Case 2f:** F is ∀xB.

**Exercise 3.5.2.**Prove (c, d, e, f).

Q.E.D.

Still, the key feature of the formulas having the form O(F) is given in

**Lemma 3.5.2**. For any formula F, there is a proof of

$$[L_1\text{-}L_9, L_{12}, L_{14}, MP, Gen]: \vdash\neg\neg O(F)\leftrightarrow O(F).$$

I.e., in the minimal logic, we may drop the double negation before O(F) (before an arbitrary formula, we can do this only in the classical logic).

**Note.** In some other textbooks, if $\neg\neg G\leftrightarrow G$ can be proved in the constructive logic, then G is called a **stable formula**. Thus, the embedding O(F) is a stable formula for any F.

**Proof.** By Theorem 2.4.4, $[L_1, L_2, L_9, MP]: \vdash A\to\neg\neg A$. Thus, it remains to prove $\vdash\neg\neg O(F)\to O(F)$. Let us proceed by induction.

**1. Induction base**: F is an atomic formula. Then O(F) is $\neg\neg F$, and $\neg\neg O(F)\to O(F)$ is $\neg\neg\neg\neg F\to\neg\neg F$. Let us remind Theorem 2.4.5: $[L_1\text{-}L_9, MP]\vdash$

¬¬A↔¬A. Hence, by taking A = ¬F:

[$L_1$-$L_9$, MP] ⊢¬¬¬¬F→¬¬F, i.e. [$L_1$-$L_9$, MP] ⊢¬¬O(F)→O(F).

**2. Induction step.**

**Case 2a**: F is $B \vee C$ , or ∃xB, or ¬B. Then O(F) is $\neg\neg(O(B) \vee O(C))$ , or ¬¬∃xO(B), or ¬O(B). Hence, ¬¬O(F)→O(F) is ¬¬¬G→¬G, where G is $\neg(O(B) \vee O(C))$ , or ¬∃xO(B), or O(B). Let us remind Theorem 2.4.5: [$L_1$-$L_9$, MP] ⊢¬¬¬A↔¬A. Hence,

[$L_1$-$L_9$, MP] ⊢¬¬¬G→¬G, i.e. [$L_1$-$L_9$, MP] ⊢¬¬O(F)→O(F).

**Case 2b:** F is B→C. Then O(F) is O(B)→O(C). By induction assumption,

[$L_1$, $L_2$, $L_{12}$, $L_{14}$, MP, Gen]: ⊢¬¬O(B)→O(B), and ⊢¬¬O(C)→O(C).

| (1) | ¬¬O(C)→O(C) | Induction assumption. |
|---|---|---|
| (2) | ¬¬(O(B)→O(C)) | ¬¬O(F) – hypothesis. |
| (3) | ¬¬O(B)→¬¬O(C) | By Theorem 2.4.7(b): [$L_1$-$L_9$, MP] ⊢ ¬¬(A→B)→(¬¬A→¬¬B). |
| (4) | O(B)→¬¬O(B) | By Theorem 2.4.4, [$L_1$, $L_2$, $L_9$, MP]: ⊢A→¬¬A. |
| (5) | O(B)→O(C), i.e. O(F) | From (4), (3) and (1), by transitivity of implication [$L_1$, $L_2$, MP]. |

Hence, since Gen is not applied here at all, by Deduction Theorem 1 [$L_1$, $L_2$, MP] we obtain that [$L_1$-$L_9$, $L_{12}$, $L_{14}$, MP, Gen] ⊢¬¬O(F)→O(F).

**Case 2c:** F is $B \wedge C$ . Then O(F) is $O(B) \wedge O(C)$ . By induction assumption,

[$L_1$, $L_2$, $L_{12}$, $L_{14}$, MP, Gen]: ⊢¬¬O(B)→O(B), and ⊢¬¬O(C)→O(C).

| (1) | $\neg\neg(O(B) \wedge O(C))$ | ¬¬O(F) – hypothesis. |
|---|---|---|
| (2) | $\neg\neg O(B) \wedge \neg\neg O(C)$ | From (1), by Theorem 2.4.8(a), [$L_1$-$L_9$, MP] ⊢ $\neg\neg(A \wedge B) \leftrightarrow (\neg\neg A \wedge \neg\neg B)$ . |
| (3) | ¬¬O(B) | From (2), by Axiom $L_3$. |
| (4) | ¬¬O(C) | From (2), by Axiom $L_4$. |

(5)  O(B)                    From (3), by induction assumption.

(6)  O(C)                    From (4), by induction assumption.

(7)  $\dfrac{O(B) \wedge O(C)}{O(F)}$ , i.e.    From (5) and (6), by Axiom $L_5$.

Hence, since Gen is not applied here at all, by Deduction Theorem 1 [$L_1$, $L_2$, MP] we obtain that [$L_1$-$L_9$, $L_{12}$, $L_{14}$, MP, Gen] $\vdash \neg\neg O(F) \rightarrow O(F)$.

**Case 2d:** F is $\forall xB$. Then O(F) is $\forall x O(B)$. By induction assumption, [$L_1$-$L_9$, $L_{12}$, $L_{14}$, MP, Gen]: $\vdash \neg\neg O(B) \rightarrow O(B)$. We must prove that $\vdash \neg\neg\forall x O(B) \rightarrow \forall x O(B)$.

(1)  $\neg\neg\forall x O(B) \rightarrow \forall x \neg\neg O(B)$    Section 3.2, I-2: [$L_1$-$L_9$, $L_{12}$, $L_{14}$, MP, Gen] $\vdash \neg\neg\forall xB \rightarrow \forall x \neg\neg B$

(2)  $\vdash \neg\neg O(B) \rightarrow O(B)$    Induction assumption

(3)  $\vdash \forall x(\neg\neg O(B) \rightarrow O(B))$    By Gen.

(4)  $\vdash \forall x \neg\neg O(B) \rightarrow \forall x O(B)$    From (3), by Theorem 3.1.1(a), [$L_1$, $L_2$, $L_{12}$, $L_{14}$, MP, Gen] $\vdash \forall x(B \rightarrow C) \rightarrow (\forall xB \rightarrow \forall xC)$.

(5)  $\vdash \neg\neg\forall x O(B) \rightarrow \forall x O(B)$    From (1) and (4), by transitivity of implication [$L_1$, $L_2$, MP].

Q.E.D.

**Lemma 3.5.3.** If F is one of the (classical) axioms $L_1$-$L_{11}$, $L_{12}$-$L_{15}$, then, in the constructive logic, [$L_1$-$L_{10}$, $L_{12}$-$L_{15}$, MP, Gen]: $\vdash O(F)$.

**Note.** The axiom $L_{10}$ will be used in the proof of Lemma 3.5.3 only once – to prove that $O(L_{10})$ is provable in the constructive logic. But, of course, $O(L_{10})$ cannot be proved in the minimal logic, hence, in the Lemma 3.5.3, the constructive logic cannot be replaced by the minimal one.

**Proof.**

**Case 1.** F (as an axiom schema) does not contain disjunctions and existential quantifiers, i.e. if F is $L_1$, $L_2$, $L_3$, $L_4$, $L_5$, $L_9$, $L_{10}$, $L_{12}$, or $L_{14}$., then O(F) is an instance of the same axiom as F, i.e. [F]: $\vdash O(F)$. For example, if F is $L_1$, i.e. $B \rightarrow (C \rightarrow B)$, then O(F) is $O(B) \rightarrow (O(C) \rightarrow O(B))$, i.e. O(F) is an instance of the same axiom $L_1$.

**Case 2a**. F is $L_6$: $B \rightarrow B \vee C$ . Then O(F) is $O(B) \rightarrow \neg\neg(O(B) \vee O(C))$ , and [[$L_1$, $L_2$, $L_6$, $L_9$, MP] $\vdash$ O(F). Indeed:

| | | |
|---|---|---|
| (1) | $O(B) \rightarrow O(B) \vee O(C)$ | Axiom $L_6$. |
| (2) | $O(B) \vee O(C) \rightarrow \neg\neg(O(B) \vee O(C))$ | By Theorem 2.4.4, [$L_1$, $L_2$, $L_9$, MP]: $\vdash A \rightarrow \neg\neg A$. |
| (3) | $O(B) \rightarrow \neg\neg(O(B) \vee O(C))$ | By transitivity of implication [$L_1$, $L_2$, MP]. |

**Case 2b**. F is $L_7$: $C \rightarrow B \vee C$ . Then O(F) is $O(C) \rightarrow \neg\neg(O(B) \vee O(C))$ , and [[$L_1$, $L_2$, $L_7$, $L_9$, MP] $\vdash$ O(F). Proof is similar to Case 2a.

**Case 2c**. F is $L_8$: $(B \rightarrow D) \rightarrow ((C \rightarrow D) \rightarrow (B \vee C \rightarrow D))$ . Then O(F) is

$$(O(B) \rightarrow O(D)) \rightarrow ((O(C) \rightarrow O(D)) \rightarrow (\neg\neg(O(B) \vee O(C)) \rightarrow O(D))) \ .$$

| | | |
|---|---|---|
| (1) | $\neg\neg O(D) \rightarrow O(D)$ | By Lemma 3.5.2, [$L_1$-$L_9$, $L_{12}$, $L_{14}$, MP,Gen]: $\vdash \neg\neg O(F) \rightarrow O(F)$. |
| (2) | O(B)→O(D) | Hypothesis. |
| (3) | (O(C)→O(D) | Hypothesis. |
| (4) | $\neg\neg(O(B) \vee O(C))$ | Hypothesis. |
| (5) | $(O(B) \rightarrow O(D)) \rightarrow ((O(C) \rightarrow O(D)) \rightarrow (O(B) \vee O(C) \rightarrow O(D)))$ . Axiom $L_8$. | |
| (6) | $O(B) v O(C) \rightarrow O(D)$ | By MP. |
| (7) | $\neg\neg(O(B) \vee O(C)) \rightarrow \neg\neg O(D)$ | From (6), by Theorem 2.4.7(a), [$L_1$-$L_9$, MP] $\vdash$ $(A \rightarrow B) \rightarrow (\neg\neg A \rightarrow \neg\neg B)$ |
| (8) | $\neg\neg O(D)$ | By MP. |
| (9) | O(D) | From (1), by MP. |

Hence, since Gen is not applied after hypotheses appear in the proof, by Deduction Theorem 2A [$L_1$, $L_2$, $L_{14}$, MP, Gen] we obtain that [$L_1$-$L_9$, $L_{12}$, $L_{14}$, MP,Gen] $\vdash$ O(F).

**Case 2d.** F is $L_{11}$: Bv¬B. Then O(F) is $\neg\neg\left(O(B)\vee\neg O(B)\right)$ . Let us remind Theorem 2.4.6(b): $[L_1\text{-}L_9, \text{MP}] \vdash \neg\neg(A\vee\neg A)$ . Hence, $[L_1\text{-}L_9, \text{MP}] \vdash$ O(F).

**Case 2e.** F is $L_{13}$: F(t)→∃xF(x). Then O(F) is O(F(t))→¬¬∃xO(F(x))), and $[[L_1, L_2, L_9, L_{13}, \text{MP}] \vdash$ O(F). Indeed:

| | | |
|---|---|---|
| (1) | O(F(t))→∃xO(F(x)) | Axiom $L_{13}$. |
| (2) | ∃xO(F(x))→¬¬∃xO(F(x)) | By Theorem 2.4.4, $[L_1, L_2, L_9, \text{MP}]$: ⊢ A→¬¬A. |
| (3) | ⊢O(F(t))→¬¬∃xO(F(x)) | By transitivity of implication $[L_1, L_2,$ MP]. |

**Case 2f.** F is $L_{15}$: ∀x(F(x)→G)→(∃xF(x)→G). Then O(F) is

$$∀x(O(F(x))→O(G))→(\neg\neg∃xO(F(x))→O(G)).$$

| | | |
|---|---|---|
| (1) | ¬¬O(G)→O(G) | By Lemma 3.5.2, $[L_1\text{-}L_9, L_{12}, L_{14},$ MP,Gen]: ⊢¬¬O(F)→O(F). |
| (2) | ∀x(O(F(x))→O(G)) | Hypothesis. |
| (3) | ¬¬∃xO(F(x)) | Hypothesis. |
| (4) | ∀x(O(F(x))→O(G))→ (∃xO(F(x))→O(G)). Axiom $L_{15}$: ∀x(F(x)→G)→(∃xF(x)→G). | |
| (5) | ∃xO(F(x))→O(G) | By MP. |
| (6) | ¬¬∃xO(F(x))→¬¬O(G) | From (4), by Theorem 2.4.7(a), $[L_1\text{-}L_9, \text{MP}]$ ⊢(A→B)→(¬¬A→¬¬B) |
| (7) | ¬¬O(G) | By MP. |
| (8) | O(G) | From (1), by MP. |

Hence, since Gen is not applied after hypotheses appear in the proof, by Deduction Theorem 2A $[L_1, L_2, L_{14}, \text{MP}, \text{Gen}]$ we obtain that $[L_1\text{-}L_9, L_{12}, L_{14}, L_{15}, \text{MP,Gen}] \vdash$ O(F).

Q.E.D.

**Theorem 3.5.4.** In the classical logic,

$$[L_1\text{-}L_{11}, L_{12}\text{-}L_{15}, MP, Gen]: B_1, B_2, ..., B_n \vdash C,$$

if and only if, in the constructive logic,

$$[L_1\text{-}L_{10}, L_{12}\text{-}L_{15}, MP, Gen]: O(B_1), O(B_2), ..., O(B_n) \vdash O(C).$$

In particular, a formula F is provable in the classical logic, if and only if the formula O(F) is provable in the constructive logic.

**Proof.**

1. Let $[L_1\text{-}L_{11}, L_{12}\text{-}L_{15}, MP, Gen]: B_1, B_2, ..., B_n \vdash C$. Induction by the length of the shortest proof.

**Induction base.** If C is an axiom, then, by Lemma 3.5.3, in the constructive logic, $\vdash O(C)$. If C is $B_i$, then $O(B_i) \vdash O(C)$ in any logic.

**Induction step.**

If C is derived by MP from B and B→C, then, by induction assumption, in the constructive logic: $O(B_1), O(B_2), ..., O(B_n) \vdash O(B)$, and $O(B_1), O(B_2), ..., O(B_n) \vdash O(B\to C)$. Let us merge these two proofs. Since $O(B\to C)$ is $O(B)\to O(C)$, then, by MP, in the constructive logic: $O(B_1), O(B_2), ..., O(B_n) \vdash O(C)$.

If C is $\forall xB(x)$, and is derived by Gen from B(x), then, by induction assumption, in the constructive logic: $O(B_1), O(B_2), ..., O(B_n) \vdash O(B(x))$. Hence, by Gen, in the constructive logic: $O(B_1), O(B_2), ..., O(B_n) \vdash \forall xO(B(x))$, i.e. $O(B_1), O(B_2), ..., O(B_n) \vdash O(F)$.

Q.E.D.

2. Let in the constructive logic: $\vdash O(B_1), O(B_2), ..., O(B_n) \vdash O(C)$. By Lemma 3.5.1, in the classical logic, $\vdash B_i\to O(B_i)$ for all i, and $\vdash O(C)\to C$. Hence, in the classical logic, $B_1, B_2, ..., B_n \vdash C$.

Q.E.D.

**Corollary 3.5.5.** If, in the classical logic, $B_1, B_2, ..., B_n \vdash C\land\neg C$, then, in the constructive logic, $O(B_1), O(B_2), ..., O(B_n) \vdash O(C)\land\neg O(C)$. I.e., if the postulates $B_1, B_2, ..., B_n$ are inconsistent in the classical logic, then the postulates $O(B_1), O(B_2), ..., O(B_n)$ are inconsistent in the constructive logic. Or: if the postulates $O(B_1), O(B_2), ..., O(B_n)$ are consistent in the constructive logic, then the postulates $B_1, B_2, ..., B_n$ are consistent in the classical logic.

**Corollary 3.5.6.** If, for some predicate language, the classical logic is inconsistent, then so is the constructive logic. Or: **if, for some predicate language, the constructive logic is consistent, then so is the classical logic** (Gödel [1933], Gentzen [1936]).

**Warning!** Corollary 3.5.6 does not extend immediately to first order **theories**, having their own specific non-logical axioms. It must be verified separately for each theory! For example,

**Exercise 3.5.3** (optional, for smart students). Verify that, if the constructive first order arithmetic is consistent, then so is the classical first order arithmetic (Gödel [1933], Gentzen [1936]). (Hint: verify that, a) atomic formulas of arithmetic are stable – this is the hard part of the proof, b) if F is an axiom of arithmetic, then so is O(F).)

Thus, the **non-constructivity does not add contradictions (at least) to arithmetic**. If it would, then we could derive "constructive" arithmetical contradictions as well.

**K. Gödel**. Zur intuitionistischen Arithmetik und Zahlentheorie. *Ergebnisse eines mathematischen Kolloquiums*, 1933, Vol. 4, pp. 34-38.

**Gerhard Gentzen**. Die Widerspruchsfreiheit der reinen Zahlentheorie. *Mathematische Annalen*, 1936, Vol. 112, pp. 493-565.

About constructive embedding operations as a general notion see

**Nikolai A.Shanin**. Embedding the classical logical-arithmetical calculus into the constructive logical-arithmetical calculus. *Dokladi AN SSSR*, 1954, vol. 94, N2, pp.193-196 (in Russian).

# 4. Completeness Theorems (Model Theory)

## 4.1. Interpretations and Models

In principle, to do the so-called pure mathematics, i.e. simply to prove theorems, one needs only **"syntax"** – axioms and rules of inference. And, for **computers**, this is the only way of doing mathematics!

But how about **"semantics"** – about our intended "vision of our domain of interest" from which we started designing of our specific predicate language and formulating axioms? First of all, we must understand that there is no way of formulating (in the predicate language and axioms) of *all the features* of our domain of interest.

For example, what information can be derived about the person *Britney* from our "theory for people" of Section 1.3? All we can derive will be formulated in terms of the following predicates:

Male(x) − means "x is a male";
Female(x) − means "x is a female";
Mother(x, y) − means "x is mother of y";
Father(x, y) − means "x is father of y";
Married(x, y) − means "x and y are married";
x=y.

Thus, there is no way of obtaining from such a theory of any information that can't be formulated in these terms, for example, about the age, colour of eyes etc.

And thus, by communicating to a computer of our predicate language and our axioms (in this language!), we have communicated only a small part of all the features of our domain of interest.

**Hence, neither our predicate language, nor our axioms can specify our initial "vision" completely. And, if so – in principle, one can imagine *many different* "visions" behind our language and axioms!**

It may seem that if, instead of our "people's domain" that is very rich in details, we will consider, for example, natural numbers, then the situation will become better, and we will be able to describe our informal "vision" unambiguously?

For example, let us considered the language of first order arithmetic (language primitives: x, y, ..., 0, 1 +, *, =), and the following non-usual "Boolean vision" B behind it.

a) As the domain of B (the "target" set of objects), instead of the set of all natural numbers, let us consider the set of "Boolean values" $D_B = \{t, f\}$. Thus, now, the variables x, y, ... can take only values $t, f$.

b) The interpretation mapping $int_B$ assigns: to the object constant 1 – the object $t$, to the object constant 0 – the object $t$, thus: $int_B(0)=f$, $int_B(1)=t$.

c) To the function constant "+" we assign the well-known *disjunction* truth table: $int_B(+) = {}'\vee'$ , to the function constant "*" – the well-known *conjunction* truth table: $int_B(*) = {}'\wedge'$ .

d) To the predicate constant "=" – the equality predicate for the set $D_B$, i.e. $int_B(=) = \{(t, t) , (f, f)\}$.

Is this vision "worse" than the usual one involving "real" natural numbers? It seems, it is worse, because the following axiom of arithmetic:

$$x+1=y+1 \rightarrow x=y$$

is false under this vision. Indeed, set x=0 and y=1: $0+1=1+1 \rightarrow 0=1$. Here, the premise is true: $0+1 = f \vee t = t = 1, 1+1 = t \vee t = t = 1$ , but the conclusion is false: 0=1 means $f=t$.

On the other hand, the following theorem of Boolean algebra:

$$x+x=x$$

is true under the above "Boolean vision" ( $t \vee t = t, f \vee f = f$ ), but it is false under the usual vision involving natural numbers.

Thus, if two theories share the same language (as do Boolean algebra and first order arithmetic), then the "validity" of a vision may depend on the formulas (axioms and theorems) that we expect to be true. If we consider only the **language**, then many different and even strange interpretations-visions will be possible. But if we consider a **theory** (i.e. language plus some specific non-logical axioms), then only a part of the interpretations-visions will be valid – only those ones, under which the specific axioms of our theory will be true. Such interpretations-visions are called **models** of the theory.

**Another example**: in our "language for people" we used names of people (*Britney, John, Paris, Peter, ...*) as object constants and the following predicate constants:

Male(x) − means "x is a male";
Female(x) − means "x is a female";
Mother(x, y) − means "x is mother of y";
Father(x, y) − means "x is father of y";
Married(x, y) − means "x and y are married";
x=y − means "x an y are the same person".

Now, let us fix the list of 4 names: *Britney, John, Paris, Peter*, and let us consider the following interpretation J of the language – a specific "small world":

a) The domain of interpretation – and the range of variables – is $D_J$ = {br, jo, pa, pe} (4 character strings).

b) int$_J$(Britney)=br, int$_J$(John)=jo, int$_J$(Paris)=pa, int$_J$(Peter)=pe.

c) int$_J$(Male) = {jo, pe}; int$_J$(Female) = {br, pa}.

d) int$_J$(Mother) = {(pa, br), (pa, jo)}; int$_J$(Father) = {(pe, jo), (pe, br)}.

e) int$_J$(Married) = {(pa, pe), (pe, pa)}.

f) int$_J$(=) = {(br, br), (jo, jo), (pa, pa), (pe, pe)}.

An alternative way of specifying interpretations of predicate constants are truth tables, for example:

| x | Male(x) | Female(x) |
|---|---------|-----------|
| br | false | true |
| jo | true | false |
| pa | false | true |
| pe | true | false |

| x | y | Father(x, y) | Mother(x, y) | Married(x, y) | x=y |
|---|---|--------------|--------------|---------------|-----|
| br | br | false | false | false | true |
| br | jo | false | false | false | false |
| br | pa | false | false | false | false |
| br | pe | false | false | false | false |
| jo | br | false | false | false | false |
| jo | jo | false | false | false | true |
| jo | pa | false | false | false | false |
| jo | pe | false | false | false | false |
| pa | br | false | true | false | false |
| pa | jo | false | true | false | false |
| pa | pa | false | false | false | true |
| pa | pe | false | true | true | false |

| | | | | | |
|---|---|---|---|---|---|
| pe | br | true | false | false | false |
| pe | jo | true | false | false | false |
| pe | pa | false | false | true | false |
| pe | pe | false | false | false | true |

Under this interpretation ("in this small world") it is true that "all people are either males, or females (but not both)", i.e., under this interpretation the corresponding formulas $\forall x(Male(x) \lor Female(x))$ and $\forall x \neg(Male(x) \land Female(x))$ qualify as true.

Under this interpretation, it is true that, "mothers are females", and that "all fathers are married people" (under this interpretation, not in the real world!). Thus, under this interpretation, the corresponding formulas ∀x(Mother(x)→Female(x)) and ∀x(Father(x)→∃y Married(x, y)) qualify as true.

But, under this interpretation ("in this small world"), it **not** true that "each person possess a mother". The corresponding formula ∀x∃y Mother(y, x) qualifies as false. As the consequence, we will see later that this formula cannot be derived from the axioms that qualify as true under this interpretation.

**Exercise 4.1.0.** Build another interpretation (a "crazy" one!) of our "4 people language", under which the following formulas are true: "some people are both male and female", "there are sex-less people", "a person may marry herself", "a person may be mother of herself".

By introducing **specific non-logical axioms**, i.e. by introducing "4 people theory" instead of pure axiom-less "4 people language" we can disqualify your "crazy" interpretation of Exercise 4.1.0. For example, the following axioms are false under it:

$$\forall x(Male(x) \lor Female(x)); \forall x \neg(Male(x) \land Female(x)).$$

### Model theory

Could the notion of "arbitrary vision" be defined precisely? For a some particular predicate language and some particular set of axioms – is there **only one** "vision" possible? Trying to answer these questions, we arrive at the so-called model theory.

**Model theory** is a very specific approach to investigation of formal theories. Model theory is using (up to) the full power of set theory. **In model theory, we investigate formal theories by using set theory as a meta-theory.**

**Paul Bernays**, in 1958: "As Bernays remarks, syntax is a branch of number theory and semantics the one of set theory." See p. 470 of

**Hao Wang**. EIGHTY YEARS OF FOUNDATIONAL STUDIES. *Dialectica,* Vol. 12, Issue 3-4, pp. 466-497, December 1958 (available online at Blackwell Synergy).

In Sections 4.1-4.3 we will develop **model theory for the classical logic**, and in Section 4.4 – model theory for the constructive propositional logic.

In the classical model theory, we will replace our vague "visions" by well-defined mathematical structures – the so-called **interpretations**. As we will see, interpretations are allowed to be non-constructive.

Technically, an interpretation will be a well-defined way of assigning "precise meanings" to all formulas of a predicate language. Any particular predicate language allows multiple ways of assigning "precise meanings" to its formulas – *multiple interpretations*.

**Interpretation of a language – the specific part**

Let L be a predicate language containing object constants $c_1$, ..., $c_k$, ... , function constants $f_1$, ..., $f_m$, ..., and predicate constants $p_1$, ..., $p_n$, .... An interpretation J of the language L consists of the following two entities:

a) A **non-empty** set $D_J$ – the domain of interpretation (it will serve first of all as the range of object variables). (Your favorite set theory comes in here.)

b) A mapping $int_J$ that assigns:

- to each object constant $c_i$ – a member $int_J(c_i)$ of the domain $D_J$ (thus, object constants "denote" particular objects in $D_J$),

- to each function constant $f_i$ – a function $int_J(f_i)$ from $D_J$ x ... x $D_J$ into $D_J$ (of course, $int_J(f_i)$ has the same number of arguments as $f_i$),

- to each predicate constant $p_i$ – a predicate $int_J(p_i)$ on $D_J$, i.e. a subset of $D_J$ x ... x $D_J$ (of course, $int_J(p_i)$ has the same number of arguments as $p_i$).

Thus, in a sense, the mapping $int_J$ assigns "meaning" to the language primitives.

The most popular example – let us consider the so-called **standard interpretation** S of first order (Peano) arithmetic PA:

a) The domain is $D_S = \{0, 1, 2, ...\}$ − the set of all natural numbers "as we know it" (more precisely – as you define it in your favorite set theory).

b) The mapping $int_S$ assigns: to the object constant 0 – the number 0, to the object constant 1 – the number 1, to the function constant "+" − the function x+y (addition of natural numbers), to the function constant "*" − the function x*y (multiplication of natural numbers), to the predicate constant "=" − the

predicate x=y (equality of natural numbers).

Yet another interpretation $J_1$ of the same language:

a) The domain is $D_{J1}$ = {e, a, aa, aaa, ...} − the set of all strings built of the letter "a" (e is the empty string).

b) The mapping $\text{int}_{J1}$ assigns: to the object constant 0 – the empty string e, to the object constant 1 – the string "a", to the function constant "+" − the concatenation function of strings, to the function constant "*" − y times concatenation of x, to the predicate constant "=" − the string equality predicate.

Yet another interpretation $J_2$ (there is no way to disqualify it as a formally correct interpretation of the language):

a) The domain is $D_{J2}$ = {$o$} – a single object $o$.

b) The mapping $\text{int}_{J2}$ assigns: to the object constant 0 – the object $o$, to the object constant 1 – the same object $o$, to the function constant "+" − the only possible function f(o,o)=o, to the function constant "*" − the only possible function f(o,o)=o, to the predicate constant "=" − the predicate {(o, o)}.

Some time later, we will use specific non-logical **axioms** to disqualify (at least some of) such "inadequate" interpretations.

Having an interpretation J of the language L, we can define the notion of **true formulas** (more precisely − the notion of formulas that are **true under the interpretation** J).

As the first step, **terms** of the language L are interpreted as members of $D_J$ or functions over $D_J$. Indeed, terms are defined as object constants, or object variables, or their combinations by means of function constants. The term $c_i$ is interpreted as the member $\text{int}_J(c_i)$ of $D_J$. The variable $x_i$ is interpreted as the function $X_i(x_i) = x_i$. And, if $t = f_i(t_1, ..., t_q)$, then $\text{int}_J(t)$ is defined as the function obtained by substituting of functions $\text{int}_J(t_1)$, ..., $\text{int}_J(t_q)$ into the function $\text{int}_J(f_i)$.

For example (first order arithmetic), the standard interpretation of the term (1+1)+1 is the number 3, the interpretation of (x+y+1)*(x+y+1) is the function $(x+y+1)^2$.

**Important − non-constructivity!** Note that, for an infinite domain $D_J$, the interpretations of function constants may be **non-computable** functions. However, if they are all computable, then we can compute the "value" of any term $t$ for any combination of values of variables appearing in $t$.

As the next step, the notion of **true atomic formulas** is defined. Of course, if a formula contains variables (as, for example, the formula x+y=1), then its

"truth-value" must be defined for each combination of values of these variables. Thus, to obtain the truth-value of the formula $p_i(t_1, ..., t_q)$ for some fixed values of the variables contained in $t_1, .., t_q$, we must first "compute" the values of these terms, and then substitute these values into the predicate $int_J(p_i)$.

For example (first order arithmetic), under the standard interpretation S, the formula x+y=1 will be true, if and only if either x takes the value 0, and y takes the value 1, or x takes the value 1, and y takes the value 0. Otherwise, the formula is false.

**Important − non-constructivity!** Note that, for an infinite domain $D_J$, the interpretations of predicate constants may be **non-computable** predicates. However, if they were all computable, then we could compute the "truth value" of any atomic formula F for any combination of values of variables appearing in F.


**Interpretations of languages − the standard common part**

And finally, we define the notion of **true compound formulas** of the language L under the interpretation J (of course, for a fixed combination of values of their free variables):

a) Truth-values of the formulas $\neg B, B \wedge C, B \vee C, B \rightarrow C$ must be computed from the truth-values of B and C (by using the well-known **classical** truth tables – see [Section 4.2](#) below).

b) The formula $\forall x\, B$ is true under J, if and only if B(c) is true under J for all members c of the domain $D_J$.

c) The formula $\exists x\, B$ is true under J, if and only if there is a member c of the domain $D_J$ such that B(c) is true under J.

For example (first order arithmetic), the formula

$$\exists y \big((x = y + y) \vee (x = y + y + 1)\big)$$

says that "x is even or odd". Under the standard interpretation of arithmetic, this formula is true for all values of its free variable x. Similarly, $\forall x\, \forall y (x + y = y + x)$ is a closed formula that is true under this interpretation.

In this way, the notion "**a closed formula F is true under the interpretation J**" is now precisely defined.

**Attention!** What about a formula F(x) containing x as free variable? In principle, under the interpretation J, it may be true for some objects, and false

– for some other objects. So, if some formula **contains free variables**, let us say that it is **true** under the interpretation J, if and only if it is true **for all combinations** of the values of its free variables.

**Important − non-constructivity!** It may seem that, under an interpretation, any closed formula is "either true or false". However, note that, for an infinite domain $D_J$, the notion of "true formulas under J" is extremely **non-constructive**: to establish, for example, the truth-value of the formula $\forall xB$, or the formula $\forall x\forall y(x+y=y+x)$, we must verify the truth of B(c) for infinitely many values of c (or a+b=b+a for infinitely many values of a and b). Of, course, this verification cannot be performed on a computer. It can only (sometimes) be proved... in some other theory. The "degree of constructivity" of the formulas like as $\forall x\exists yC(x,y)$, $\forall x\exists y\forall zD(x,y,z)$ etc. is even less than that...

**Empty Domains?** Do you think, we should consider also **empty domains** of interpretation? According to the axiom $L_{13}$: $(B{\to}B){\to}\exists x(B{\to}B)$, hence, $\exists x(B{\to}B)$. In an empty domain, this formula would be false. Thus, to cover the case of empty domain, we must modify our axioms of the predicate logic. In this way one obtains the so-called **free logic**. For details, see Free logic in Wikipedia, and Section 2.16 in Mendelson [1997].

**Three kinds of formulas**

If one explores some formula F of the language L in various interpretations, then three situations are possible:

a) F is **true in all** interpretations of the language L. Formulas of this kind are called **logically valid formulas**.

b) F is **true in some** interpretations of L, and **false − in some** other interpretations of L.

c) F is **false in all** interpretations of L (then, of course, ¬F is true in all interpretations). Formulas of this kind are called **unsatisfiable formulas.**

Formulas that are "not unsatisfiable" (i.e. formulas of classes (a) and (b)) are called, of course, **satisfiable formulas**: a formula is satisfiable, if it is true in at least one interpretation.

**Note**. These definitions are pretty clear for a **close** formula F. However, remember that, if some formula **contains free variables**, we say that it is **true** under the interpretation J, if and only if it is true **for all combinations** of the values of its free variables. This convention allows to apply the above definitions also to the formulas containing free variables.

**Exercise 4.1.1.** Verify that: a) **F is satisfiable, if and only if ¬F is not logically valid**. b) **F is logically valid, if and only if ¬F is unsatisfiable**.

**Logically valid formulas**

Some formulas are true under all interpretations, for example:

$$(B \rightarrow C) \wedge (C \rightarrow D) \rightarrow (B \rightarrow D) \quad,$$
$$F(x) \rightarrow \exists x F(x) \quad,$$
$$\forall x F(x) \rightarrow F(x) \quad,$$
$$\forall x (F(x) \rightarrow G(x)) \rightarrow (\forall x F(x) \rightarrow \forall x G(x)) \quad,$$
$$\forall x (F(x) \rightarrow G(x)) \rightarrow (\exists x F(x) \rightarrow \exists x G(x)) \quad,$$
$$\forall x (G(x) \wedge H(x)) \rightarrow (\forall x G(x) \wedge \forall x H(x)) \quad,$$
$$\exists x (G(x) \vee H(x)) \rightarrow (\exists x G(x) \vee \exists x H(x)) \quad.$$

How about the axioms $L_1$-$L_{15}$?

Such formulas are called **logically valid**. More precisely, in a predicate language L, a formula is called **logically valid**, if and only if it is true in **all interpretations** of the language L for all values of its free variables.

Thus, a logically valid formula is true independently of its "meaning" − the interpretations of constants, functions and predicates used in it. But note that here, the (classical!) **interpretations of propositional connectives and quantifiers remain fixed.**

In a sense, logically valid formulas are "content-free": they do not give us any specific information about features of objects they are "speaking" about.

**Important − non-constructivity!** The notion of logically valid formulas is **doubly non-constructive** in the sense that the universal quantifier "for all interpretations" is added to the (already) non-constructive definition of a true formula.

As we will see later, all the axioms of our classical logical axiom system [$L_1$-$L_{15}$, MP, Gen] are logically valid formulas. And it appears that, from logically valid formulas, the inference rules MP and Gen generate only logically valid formulas. I.e. we will prove that **all the formulas that can be proved in the classical logic [$L_1$-$L_{15}$, MP, Gen], are logically valid (the so-called soundness theorem of the classical predicate logic).**

As an example, let us verify that the axiom $L_{12}$: $\forall x F(x) \rightarrow F(t)$ is logically valid. Let us assume the contrary, i.e. that, under some interpretation J, for some values of its free variables, $L_{12}$ is false. According to the classical truth tables, this could be only, if and only if $\forall x F(x)$ were true, and $F(t)$ were false (under the interpretation J, for the same above-mentioned values of free

variables). Let us "compute" the value of the term t for these values of free variables (since the substitution F(x/t) is admissible, t may contain only these variables), and denote it by c. Thus, F(c) is false. But ∀xF(x) is true, hence, F(a) is true for all a in the domain $D_J$, i.e. F(c) also is true. Contradiction. Hence, $L_{12}$ is true under all interpretations for all combinations of its free variables (if any).

**Exercise 4.1.2.** Verify that the remaining 6 of the above formulas are logically valid. (Hint: follow the above example − assume that there is an interpretation J such that the formula under question is false for some values of its free variables, and derive a contradiction.)

**Is our axiom system of logic powerful enough to prove ALL the logically valid formulas?** The answer is positive − see **Gödel's Completeness Theorem** in : a formula is logically valid, if and only if it is provable in the **classical** logic [$L_1$-$L_{11}$, $L_{12}$-$L_{15}$, MP, Gen].

But, of course, there are formulas that are not logically valid. For example, negations of logically valid formulas are false in all interpretations, i.e. they are not logically valid. Such formulas are called **unsatisfiable formulas**. But there are formulas that are true in some interpretations, and false − in some other ones. An example of such formulas: the axiom of arithmetic x+1=y+1 → x=y considered above. It is true under the standard interpretation, but it can be made false by "crazy" interpretations of the equality predicate (for example, by setting 1=1 as true, but 0=0 – as false).

To conclude that some formula is **not** logically valid, we must build an interpretation J such that the formula under question is false for some values of its free variables.

As an example, let us verify that the formula

$$\forall x(p(x) \vee q(x)) \rightarrow \forall x\, p(x) \vee \forall x\, q(x)$$

is not logically valid (p, q are predicate constants). Why it is not? Because the truth-values of p(x) and q(x) may behave in such a way that $p(x) \vee q(x)$ is always true, but neither ∀x p(x), nor ∀x q(x) is true. Indeed, let us take the domain D = {a, b}, and set:

| x | p(x) | q(x) |
|---|------|------|
| a | true | false |
| b | false | true |

In this interpretation, $p(a) \vee q(a)$ = true, $p(b) \vee q(b)$ = true, i.e. the premise $\forall x(p(x) \vee q(x))$ is true. But the formulas ∀x p(x), ∀x q(x) both

are false. Hence, in this interpretation, the consequent $\forall x\, p(x) \vee \forall x\, q(x)$ is false, and thus, $\forall x(p(x) \vee q(x)) \to \forall x\, p(x) \vee \forall x\, q(x)$ is false. We have built an interpretation, making false the formula under question. Q.E.D.

On the other hand, this formula **is satisfiable** – there is an interpretation under which it is true. Indeed, let us take D={a} as the domain of interpretation, and let us set p(a)=q(a)=true. Then all the formulas

$$\forall x(p(x) \vee q(x)), \; \forall x\, p(x), \; \forall x\, q(x)$$

become true, and so is the entire formula under consideration. Q.E.D.

**Exercise 4.1.3.** Verify that the following formulas are satisfiable, but **not** logically valid (p, q, r are predicate constants):

a) $p(x,y) \wedge p(y,z) \to p(x,z)$ ,

b) q(x)→∀x q(x),

c) (∀x q(x)→∀x r(x))→∀x(q(x)→r(x)),

$c_1$) ∃x(p(x)→B)→(∃x p(x)→B), where B does not contain x,

d) ∀x∃y p(x, y)→∃y∀x p(x, y),

e) $\exists x\, q(x) \wedge \exists x\, r(x) \to \exists x(q(x) \wedge r(x))$ ,

f) $\forall x \neg p(x,x) \wedge \forall x\, \forall y\, \forall z(p(x,y) \wedge p(y,z) \to p(x,z)) \to$
$$\forall x\, \forall y(x=y \vee p(x,y) \vee p(y,x)) \;.$$

Hint. For the domain D={a, b}, it is convenient to use tables to define your interpretation of binary predicate letters, for example,

| x | y | r(x, y) |
|---|---|---------|
| a | a | false |
| a | b | true |
| b | a | true |
| b | b | false |

**Exercise 4.1.4.** Is the following formula logically valid, or not (p, q are predicate constants):

$$(\exists x\, p(x) \to \exists x\, q(x)) \to \exists x(p(x) \to q(x)).$$

(Hint: follow the above example − use natural numbers or other objects trying to build an interpretation J such that the formula under question is false.)

**Satisfiability**

We already know that, in a predicate language L, a formula F is called **satisfiable**, if and only if **there is an interpretation** of the language L such that F is true for **some** values of its free variables (we will say also that F **is satisfied under this interpretation**). A set of formulas $F_1$, ..., $F_n$, ... is called satisfiable, if and only if there is an interpretation under which the formulas $F_1$, ..., $F_n$, ... are satisfied *simultaneously*.

**Examples**. a) Formula $\exists x\ p(x)$ is, of course, not logically valid, but it is satisfiable, because it is true in the following interpretation J: $D_J=\{b\}$, p(b) is true.

b) Formulas x*0=0, x+y=y+x and x+(y+z)=(x+y)+z are **not** logically valid (see Exercise 4.1.7 below), but they are satisfiable, because they are true under the standard interpretation of arithmetic.

**Exercise 4.1.5.** a) Verify that the formula $\forall xy(p(x)\rightarrow p(y))$ is true in all one-element interpretations (i.e. when the interpretation domain consists of a single element), but is false in at least one two-element interpretation (p is a predicate constant).

b) Verify that the formula

$$\forall x\ \forall y\ \forall z[(p(x)\leftrightarrow p(y))\vee(q(y)\leftrightarrow q(z))\vee(r(z)\leftrightarrow r(x))]$$

is true in all one- and two-element interpretations, but is false in at least one three-element interpretation (p, q, r are predicate constants).

c) Prove that the formula $\exists x\forall y\ F(x,y)$ is logically valid, if and only if so is the formula $\exists x\ F(x, g(x))$, where g is a function constant that does not appear in F.

d) Prove that the formula $\forall x\forall y\exists z\ F(x,y,z)$ is satisfiable, if and only if so is the formula $\forall x\forall y\ F(x, y, h(x,y))$, where h is a function constant that does not appear in F.

**Logical consequences**

"$A_1$, ..., $A_n$ imply G", or "the formula G follows from the formulas $A_1$, ..., $A_n$" − what should this mean in general? If $A_1$, ..., $A_n$ are true, then G is true? Always, or under some specific conditions? Assume, that **all** these conditions are specified in the formulas $A_1$, ..., $A_n$. Then, G follows from $A_1$, ..., $A_n$ unconditionally ("logically"), i.e. if the formulas $A_1$, ..., $A_n$ all true, then G must be true without any other conditions. Since the notion of "true" we have

formalized as "true under an interpretation", we can formalize the notion of "logical consequence" as follows:

G is a **logical consequence** of $A_1$, ..., $A_n$, if and only if **G is true under any interpretation, under which $A_1$, ..., $A_n$ are all true**.

Or, equivalently:

G is a **logical consequence** of $A_1$, ..., $A_n$, if and only if **G is true in any model of $A_1$, ..., $A_n$**.

**Exercise 4.1.6**. Verify three fundamental "facts":

**a) The formula G is a logical consequence of the formulas $A_1$, ..., $A_n$, if and only if the formula**

$$A_1 \wedge ... \wedge A_n \rightarrow G$$

**is logically valid.**

**b) If the set of formulas $A_1$, ..., $A_n$ is satisfiable, then the formulas B, ¬B cannot both be logical consequences of $A_1$, ..., $A_n$.**

**c) The formula G is a logical consequence of the formulas $A_1$, ..., $A_n$, if and only if the set $A_1$, ..., $A_n$, ¬G is unsatisfiable.**

In [Section 4.3](#) we will prove Gödel's Completeness Theorem. It will imply that G is a logical consequence of $A_1$, ..., $A_n$, if and only if

$$[L_1\text{-}L_{11}, L_{12}\text{-}L_{15}, MP, Gen]: \quad A_1 \wedge ... \wedge A_n \vdash G,$$

i.e. if the formula G can be derived from the formulas $A_1$, ..., $A_n$ by using the classical logic.


**Theories and their models**

If T is a first order theory, and J is an interpretation of its language, and if J makes true the specific axioms of T, then (as we already know) J is called a **model of** T.

For non-mathematical people, this term "model of a theory" may seem somewhat strange: in other branches of science, theories serve as a basis for building models of natural phenomena, of technical devices etc. But only the term is strange ("upside down") here, the process is the same as in other branches of science: first order theories "generate" their models, and these models can be used for modeling of natural phenomena, of technical devices etc.

**Warning. Specific axioms of a first order theory T are not logically valid formulas!** They are not true in all interpretations, they are true **only in the models** of T. Models of T − it is a **proper subclass** of all the possible interpretations. For example, the "obvious" arithmetical axioms like as x+0=x (or, theorems like as x+y=y+x) are not logically valid. If we would interpret 0 as the number "two", then x+0 and x will be equal! Logically valid formulas must be true under **all** interpretations!

**Exercise 4.1.7.** Verify that x=x, x*0=0, x+y=y+x and x+(y+z)=(x+y)+z are satisfiable, but not logically valid formulas.

As we already noted above, in a sense, **logically valid formulas "do not contain information"** (are "content-free") − just because they are true in all interpretations, i.e. they are true independently of the "meaning" of language primitives. Indeed, let us consider the formulas x+0=x → x+0=x, and x+0=0 → x+0=0. Both are logically valid, but do we get more information about zero and addition after reading them? Another example: 2*2=5 → 2*2=5, or 2*2=4 → 2*2=4, these formulas also are logically valid, but do they help in computing the value of 2*2? The specific axioms of some theory T, on the contrary, do "contain information" − they separate a proper subclass of all interpretations − models of T.

Do the axioms of first order arithmetic "specify" the standard interpretation S, i.e. are the axioms of first order arithmetic true **in this interpretation only?** No, there are many **non-standard interpretations** making these axioms true! More: see Non-standard model of arithmetic in Wikipedia.

### Transitive predicates and recursion

Let us return to the problem that we considered already in Section 1.2.

How about the predicate **Ancestor(x, y)** − "x is an ancestor of y"? Could it be expressed as a formula of our "language for people"? The first idea − let us "define" this predicate recursively:

$$\forall x \, \forall y \, (Father(x, y) \vee Mother(x, y) \rightarrow Ancestor(x, y)) \; ;$$
$$\forall x \, \forall y \, \forall z \, (Ancestor(x, y) \wedge Ancestor(y, z) \rightarrow Ancestor(x, z)) \; .$$

The second rule declares the transitivity property of the predicate. The above two formulas are **axioms**, allowing to derive essential properties of the predicate Ancestor(x, y). But how about a single formula F(x, y) in the "language for people", expressing that "x is an ancestor of y"? Such a formula should be a tricky combination of formulas Father(x, y), Mother(x, y) and x=y. And such a formula is **impossible**! See Transitive closure in Wikipedia, and Theorem 1.2 and its proof in:

**Carlos Areces**. Logic Engineering. The Case of Description and Hybrid Logics. Ph.D. Thesis,

University of Amsterdam, Amsterdam, The Netherlands, 2000.

**Exercise 4.1.8** (optional, for smart students). Explain the precise meaning of the statement: in the "language for people", formula F(x, y) expresses that "x is an ancestor of y".

## 4.2. Classical Propositional Logic − Truth Tables

[Emil Leon Post](#) (1897-1954): "... Post's Ph.D. thesis, in which he proved the completeness and consistency of the propositional calculus described in the *Principia Mathematica* by introducing the truth table method. He then generalised his truth table method, which was based on the two values "true" and "false", to a method which had an arbitrary finite number of truth-values." (According to [MacTutor History of Mathematics archive](#)).

First, let us consider the **classical propositional logic**. Here, each formula is built of some is built of some "atoms" $B_1$, $B_2$, ..., $B_n$ by using propositional connectives only (i.e. $B \land C$, $B \lor C$, $\neg B$, $B \to C$ ). Our axioms for this logic we represented as axiom schemata $L_1$-$L_{11}$, in which the letters B, C, D can be replaced by any formulas.

Is our list $L_1$-$L_{11}$ of classical propositional axiom schemata "complete"? Aren't some necessary axiom schemata missing there? If something necessary is missing, we must add it to the list.

This problem was solved by [Emil L. Post](#) in 1920. He proved that **if one would add to $L_1$-$L_{11}$ as an axiom schema any formula that can't be proved from these axioms, then one would obtain an inconsistent system in which all formulas are provable.** Thus, in fact, nothing is missing in our list of classical propositional axioms.

Post proved his theorem by using the so-called **classical truth tables**. Each propositional atom may take any of two truth-values – *true* and *false*. And, if we already know truth-values of the formulas B, C, then we can use truth tables to compute truth-values of the formulas $B \land C$, $B \lor C$, $\neg B$, $B \to C$ .

If B is false, and C is false, then $B \land C$ is false.
If B is false, and C is true, then $B \land C$ is false.
If B is true, and C is false, then $B \land C$ is false.
If B is true, and B is true, then $B \land C$ is true.

| B | C | $B \land C$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |

| | | |
|---|---|---|
| 1 | 1 | 1 |

If B is false, and C is false, then $B \vee C$ is false.
If B is false, and C is true, then $B \vee C$ is true.
If B is true, and C is false, then $B \vee C$ is true.
If B is true, and C is true, then $B \vee C$ is true.

| **B** | **C** | $B \vee C$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

If B is false, then ¬B is true.
If B is true, then ¬B is false.

| **B** | **¬B** |
|---|---|
| 0 | 1 |
| 1 | 0 |

No problems so far.

If B is false, and C is false, then B→C is what? True? False? But, why?
If B is false, and C is true, then B→C is what? True? False? But, why?
If B is true, and C is false, then B→C is false, of course.
If B is true, and C is true, then B→C is what? Perhaps, not false? Hence, true?

How to answer the 3 what's? If B is false, then B→C possesses no real meaning. And, if we already know that B is true, and C is true, then B→C is no more interesting. But, if a definite "truth-value" for B→C is mandatory in all cases, then we can *greatly simplify the situation* by assuming that B→C is true, except, if B is true, and C is false. Thus:

If B is false, and C is false, then B→C is true.
If B is false, and C is true, then B→C is true.
If B is true, and C is false, then B→C is false.
If B is true, and C is true, then B→C is true.

| **B** | **C** | **B→C** |
|---|---|---|
| 0 | 0 | 1 |

| 0 | 1 | 1 |
|---|---|---|
| 1 | 0 | 0 |
| 1 | 1 | 1 |

This definition is equivalent to saying that

B→C is true, if and only if $\neg(B \wedge \neg C)$ is true

or:

B→C is false, if and only if B is true, and C is false.

In this way, having any formula F and some assignment of truth-values to its atoms, we can compute the truth-value of F.

In fact, this setting represents a specific kind of *interpretations* – in terms of the above Section 4.1. Our propositional atoms replace the atomic formulas of predicate languages – both (atoms and formulas) are either true or false, and the truth-value of a propositional formula depends on the truth-values of its atoms only. Thus, we are considering, in fact, interpretations J with a fixed domain $D_J = \{true, false\}$, and in each interpretation, each atom possess a definite truth-value – *true* or *false*.

What would happen to some propositional formula F, if we would try *all the possible* truth-values of all the propositional atoms occurring in F? This corresponds to trying out formulas of predicate languages in all interpretations.

There are three possibilities:

F takes only true values (thus, F is logically valid);

F takes only false values (thus, F is unsatisfiable);

F takes both of values (thus, F is satisfiable, but not logically valid).

**Lemma 4.2.1.** Under the classical truth tables, all the classical propositional axioms $L_1$-$L_{11}$ take only true values.

**Proof.** First, let us verify $L_{11}$ and $L_{10}$:

| **B** | **¬B** | $B \vee \neg B$ |
|---|---|---|
| 0 | 1 | 1 |
| 1 | 0 | 1 |

| **B** | **C** | **¬B** | **B→C** | **¬B→(B→C)** |
|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 1 |

| | | | | |
|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 |

**Exercise 4.2.1.** Verify that the axioms $L_1$-$L_9$ take only true values as well.

See also:
"Truth Tables" from The Wolfram Demonstrations Project. Contributed by: **Hector Zenil**.

**Lemma 4.2.2.** If, under the classical truth tables, the formulas B and B→C take only true values, then so does C. I. e. from "always true" formulas, *Modus Ponens* allows deriving only of "always true" formulas.

**Proof.** Let us assume that, in some situation, C takes a false value. In the same situation, B and B→C take true values. If B is true, and C is false, then B→C is false. Contradiction. Hence, C takes only true values. Q.E.D.

**Note**. In the proof of Lemma 4.2.2, only the third row of implication truth table was significant: if B is true, and C is false, then B→C is false!

**Theorem 4.2.3 (soundness of the classical propositional logic).**

If $[L_1$-$L_{11}$, MP]: ⊢ F, then, under the classical truth tables, F takes only true values. In particular: the classical propositional logic is **consistent** – in the sense that one cannot prove both $[L_1$-$L_{11}$, MP]: ⊢ $G$ and $[L_1$-$L_{11}$, MP]: ⊢ $\neg G$ , for any formula G.

**Proof.** By induction, from Lemmas 4.2.1 and 4.2.2.

**Completeness of classical propositional logic**

How about the converse statement of Theorem 4.2.3: if, under the classical truth tables, formula F takes only true values, then $[L_1$-$L_{11}$, MP]: ⊢F? I.e., **are our axioms powerful enough to prove any formula that is taking only true values?** The answer is "yes":

**Theorem 4.2.4 (completeness of the classical propositional logic).** Assume, the formula F has been built of formulas $B_1$, $B_2$, ..., $B_n$ by using propositional connectives only. If, under the classical truth tables, for any truth-values of $B_1$, $B_2$, ..., $B_n$, formula F takes only true values, then:

a) in the constructive logic,

$$[L_1\text{-}L_{10}, \text{MP}]: \quad B_1 \vee \neg B_1, B_2 \vee \neg B_2, ..., B_n \vee \neg B_n \ \vdash F,$$

b) in the classical logic, [L$_1$-L$_{11}$, MP]: ⊢F.

Of course, (b) follows from (a) immediately − all the premises $B_1 \vee \neg B_1, B_2 \vee \neg B_2, ..., B_n \vee \neg B_n$ are instances of the axiom L$_{11}$.

**Corollary 4.2.4**. The classical propositional axioms [L$_1$-L$_{11}$, MP] are "complete" also in the sense that if one would add any formula that can't yet be proved from these axioms, then one would obtain a system, in which all formulas are provable, i.e. an inconsistent system.

Indeed, if some formula F can't be proved from [L$_1$-L$_{11}$, MP], then it takes false value for some combination of truth-values of its atoms. Replace each true atom by the formula A→A, and each false atom – by ¬(A→A). In this way we obtain a formula F' that takes only false values, i.e. ¬F' takes only true values, and hence, can be proved from [L$_1$-L$_{11}$, MP]. Thus, if we would add F to [L$_1$-L$_{11}$, MP] as an axiom schema, then, in this system, the formulas F' and ¬F' will be provable, and by L$_{10}$ – any formula will be provable.

**Note.** Assume, the formula F is built of atoms B$_1$, B$_2$, ..., B$_n$ by using propositional connectives only. If, under the classical truth tables, for any (possible and impossible) truth-values of B$_1$, B$_2$, ..., B$_n$, formula F takes only true values, then F is called a **tautology**. Theorem 4.2.4 says that **any tautology can be proved in the classical propositional logic**.

Completeness of the classical propositional logic was first proved by Emil L. Post in his 1920 Ph.D. thesis, and published as

**E. Post**. Introduction to a general theory of elementary propositions. *American Journal of Mathematics*, 1921, vol. 43, pp.163-185.

About the history, see also:

**Richard Zach**. Completeness before Post: Bernays, Hilbert, and the development of propositional logic. *The Bulletin of Symbolic Logic*, 1999, vol. 5, N3, pp.331-366.

Following an elegant later idea by Laszlo Kalmar we need two simple lemmas before trying to prove this theorem.

**L. Kalmar**. Über Axiomatisiebarkeit des Aussagenkalküls. *Acta scientiarium mathematicarum (Szeged)*. 1934-35. vol. 7, pp. 222-243.

**Lemma 4.2.5.** In the constructive logic, one can "compute" the classical truth-values of $\neg B, B \rightarrow C, B \wedge C, B \vee C$ in the following sense:

| Negation | Implication | Conjunction | Disjunction |
|---|---|---|---|
| [ ]:<br><br>$\neg B \vdash \neg B$ | $[L_{10}, MP]$:<br><br>$\neg B, \neg C \vdash B \rightarrow C$ | $[L_1, L_2, L_3, L_9, MP]$:<br><br>$\neg B, \neg C \vdash$<br>$\neg(B \wedge C)$ | $[L_1\text{-}L_9, MP]$:<br><br>$\neg B, \neg C \vdash$<br>$\neg(B \vee C)$ |
| $[L_1, L_2, L_9, MP]$:<br><br>$B \vdash \neg\neg B$ | $[L_{10}, MP]$:<br><br>$\neg B, C \vdash B \rightarrow C$ | $[L_1, L_2, L_3, L_9, MP]$:<br><br>$\neg B, C \vdash \neg(B \wedge C)$ | $[L_7, MP]$:<br><br>$\neg B, C \vdash B \vee C$ |
| | $[L_1, L_2, L_9, MP]$:<br><br>$B, \neg C \vdash \neg(B \rightarrow C)$ | $[L_1, L_2, L_4, L_9, MP]$:<br><br>$B, \neg C \vdash \neg(B \wedge C)$ | $[L_6, MP]$:<br><br>$B, \neg C \vdash B \vee C$ |
| | $[L_1, MP]$:<br><br>$B, C \vdash B \rightarrow C$ | $[L_5, MP]$:<br><br>$B, C \vdash B \wedge C$ | $[L_6, MP]$:<br><br>$B, C \vdash B \vee C$ |

**Note**. Thus, to "compute" the classical truth-values, the axiom $L_{11}$ is not necessary!

**Proof.**

**$\neg B \vdash \neg B$**

Immediately, in any logic.

**$B \vdash \neg \neg B$**

By Theorem 2.4.4. $[L_1, L_2, L_9, MP]$: $\vdash A \rightarrow \neg\neg A$.

**$\neg B, C \vdash B \rightarrow C$**

**$\neg B, \neg C \vdash B \rightarrow C$**

By axiom $L_{10}$: $\neg B \rightarrow (B \rightarrow C)$ we obtain $\neg B \vdash B \rightarrow C$. This covers both cases.

**$B, \neg C \vdash \neg(B \rightarrow C)$**

This is exactly Theorem 2.4.1(c) $[L_1, L_2, L_9, MP]$.

**$B, C \vdash B \rightarrow C$**

By axiom $L_1$: $C \rightarrow (B \rightarrow C)$ we obtain $C \vdash B \rightarrow C$.

**$\neg B, \neg C \vdash \neg(B \wedge C)$**

**$\neg B, C \vdash \neg(B \wedge C)$**

By axiom $L_3$: $B \wedge C \rightarrow B$ and the Contraposition Law (Theorem 2.4.2) $[L_1,$

$L_2$, $L_9$, MP]: ⊢(A→B)→(¬B→¬A) we obtain ⊢ $\neg B \rightarrow \neg (B \wedge C)$ , and ¬B ⊢ $\neg (B \wedge C)$ . This covers both cases.

**B, ¬C ⊢ $\neg (B \wedge C)$**

By axiom $L_4$: $B \wedge C \rightarrow C$ and the Contraposition Law (Theorem 2.4.2) [$L_1$, $L_2$, $L_9$, MP]: ⊢(A→B)→(¬B→¬A) we obtain ⊢ $\neg C \rightarrow \neg (B \wedge C)$ , an ¬C ⊢ $\neg (B \wedge C)$ .

**B, C ⊢ $B \wedge C$**

By axiom $L_5$: $B \rightarrow (C \rightarrow B \wedge C)$ we obtain B, C ⊢ $B \wedge C$ .

**¬B, ¬C ⊢ $\neg (B \vee C)$**

By Theorem 2.4.10(b).

**¬B, C ⊢ $B \vee C$**

By axiom $L_7$: $C \rightarrow B v C$ we obtain C ⊢ $B \vee C$ .

**B, ¬C ⊢ $B \vee C$**

**B, C ⊢ $B \vee C$**

By axiom $L_6$: $B \rightarrow B \vee C$ we obtain B ⊢ $B \vee C$ . This covers both cases.

Q.E.D.

As the next step, we will generalize Lemma 4.2.5 by showing how to "compute" truth-values of arbitrary formula F, which is built of formulas $B_1$, $B_2$, ..., $B_n$ by using more than one propositional connective. For example, let us take the formula $B \vee C \rightarrow B \wedge C$ :

| B | C | $B \vee C$ | $B \wedge C$ | $B \vee C \rightarrow B \wedge C$ |
|---|---|------------|--------------|-----------------------------------|
| 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 |

We will show that, in the constructive logic [$L_1$-$L_{10}$, MP]:

¬B, ¬C ⊢ $B \vee C \rightarrow B \wedge C$ ,
¬B, C ⊢ $\neg (B \vee C \rightarrow B \wedge C)$ ,
B, ¬C ⊢ $\neg (B \vee C \rightarrow B \wedge C)$ ,

$\neg B, \neg C \vdash B \lor C \to B \land C$ .

**Lemma 4.2.6.** Assume, the formula F has been built of formulas $B_1$, $B_2$, ..., $B_n$ by using propositional connectives only. Assume that, if the formulas $B_1$, $B_2$, ..., $B_n$ take the truth-values $v_1$, $v_2$, ..., $v_n$ respectively, then, for these values, formula F takes the truth-value w. Then, in the constructive logic, we can "compute" the truth-value of F in the following sense:

$$[L_1\text{-}L_{10}, MP]: v_1B_1, v_2B_2, ..., v_nB_n \vdash wF,$$

where: wF denotes F, if w is true, and ¬F, if w is false, and $v_iB_i$ denotes $B_i$, if $v_i$ is true, and $\neg B_i$, if $v_i$ is false.

**Proof.** By induction.

**Induction base.** F is one of the formulas $B_i$. Then $w = v_i$, and, of course, in any logic, $v_iB_i \vdash wF$.

**Induction step.**

Note that Lemma 4.2.5 represents the assertion of Lemma 4.2.6 for formulas built of $B_1$, $B_2$, ..., $B_n$ by using a single propositional connective.

1. **F is ¬G**. By the induction assumption,

$$[L_1\text{-}L_{10}, MP]: v_1B_1, v_2B_2, ..., v_nB_n \vdash w'G,$$

where w' represents the truth-value of G. By Lemma 4.2.5,

$[L_1\text{-}L_{10}, MP]: w'G \vdash wF$, hence, $[L_1\text{-}L_{10}, MP]: v_1B_1, v_2B_2, ..., v_nB_n \vdash wF$.

2. **F is G o H**, where o is implication, conjunction, or disjunction. By the induction assumption,

$$[L_1\text{-}L_{10}, MP]: v_1B_1, v_2B_2, ..., v_nB_n \vdash w'G,$$

where w' represents the truth-value of G, and

$$[L_1\text{-}L_{10}, MP]: v_1B_1, v_2B_2, ..., v_nB_n \vdash w''H,$$

where w'' represents the truth-value of H. By Lemma 4.2.5,

$$[L_1\text{-}L_{10}, MP]: w'G, w''H \vdash wF,$$

hence, $[L_1\text{-}L_{10}, MP]: v_1B_1, v_2B_2, ..., v_nB_n \vdash wF$.

Q.E.D.

**Proof of Theorem 4.2.4(a).** By Lemma 4.2.6:

$$[L_1\text{-}L_{10}, MP]: B_1, v_2B_2, ..., v_nB_n \vdash F,$$

$$[L_1\text{-}L_{10}, MP]: \neg B_1, v_2 B_2, ..., v_n B_n \vdash F,$$

because F takes only true values. By $[L_1, L_2, MP]$ Deduction Theorem 1,

$$[L_1\text{-}L_{10}, MP]: v_2 B_2, ..., v_n B_n \vdash B_1 \rightarrow F,$$
$$[L_1\text{-}L_{10}, MP]: v_2 B_2, ..., v_n B_n \vdash \neg B_1 \rightarrow F,$$

Let us merge these two proofs and append an instance of the axiom $L_8$:

$$\vdash (B_1 \rightarrow F) \rightarrow ((\neg B_1 \rightarrow F) \rightarrow (B_1 \vee \neg B_1 \rightarrow F)) \quad .$$

Hence, by MP:

$$[L_1\text{-}L_{10}, MP]: v_2 B_2, ..., v_n B_n \vdash B_1 \vee \neg B1 \rightarrow F \quad ,$$

and

$$[L_1\text{-}L_{10}, MP]: B_1 v \neg B_1, v_2 B_2, ..., v_n B_n \vdash F.$$

By repeating this operation we obtain Theorem 4.2.4(a):

$$[L_1\text{-}L_{10}, MP]: \quad B_1 \vee \neg B_1, B_2 \vee \neg B_2, ..., B_n \vee \neg B_n \quad \vdash F.$$

Q.E.D.

**Exercise 4.2.1A** (optional). The above proof of the Completeness Theorem contains, in fact, an **algorithm** allowing to build a (very long) proof of any propositional formula that takes only true values. Try computing how many formulas contains this "very long" proof of a formula containing $n$ propositional atoms.

## Consequences of the Completeness Theorem

From now on, we could forget our great ability of proving formulas in the classical propositional logic, that we learned in Section 2. Indeed, in order to verify, is a formula provable in $[L_1\text{-}L_{11}, MP]$, or not, we can simply check, under the classical truth tables, takes this formula only true values, or not. If it takes, then, by Completeness Theorem, formula is provable, if not – it is not provable. For formulas containing one, two or three atoms, this method is really simpler than direct proving of formulas in $[L_1\text{-}L_{11}, MP]$. However, we must remember that for longer formulas, this method allows to build automatically (by using Lemmas 4.2.5 and 4.2.6) only of extremely long proofs.

**Exercise 4.2.2** (optional, for smart students). Verify that.

## Computational complexity of the problem

However, note that, if the formula F contains $n$ different atoms A, B, C, ...,

then its truth table contains $2^n$ rows for which the truth value of F must be checked one by one. Of course, if the formula contains 2 atoms (like as $(A \to B) \to \neg A \lor B$, or 3 atoms (like as the Axiom $L_2$), then its truth table consists of 4 or 8 rows − for most people this is a feasible task. But the "truth table" for a formula containing 32 atoms contains four billion rows to check... So, let us try inventing a more efficient algorithm? Until now, all the attempts have produced algorithms requiring more than $2^{cn}$ units of time for checking a formula containing $n$ different atoms ($c$ is a constant depending on the algorithm).

And it seems, we will never really succeed − the problem of determining the **classical** provability of propositional formulas belongs to the so-called complexity class "co-NP-complete", see Boolean satisfiability problem in Wikipedia.

The problem of determining the **constructive** provability of propositional formulas is even harder – it belongs to the complexity class "PSPACE-complete", see:

**Richard Statman.** Intuitionistic propositional logic is polynomial-space complete, Theoretical Computer Science 9 (1979), pp. 67–72 (online copy available).

## 4.3. Classical Predicate Logic − Gödel's Completeness Theorem

Kurt Gödel (1906-1978) "He is best known for his proof of Gödel's Incompleteness Theorems. In 1931 he published these results in *Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme* . ...Gödel's results were a landmark in 20th-century mathematics, showing that mathematics is not a finished object, as had been believed. It also implies that a computer can never be programmed to answer all mathematical questions." (According to MacTutor History of Mathematics archive).

As David Hilbert and Wilhelm Ackermann published in

**D. Hilbert, W. Ackermann**. Grundzüge der theoretischen Logik. Berlin (Springer), 1928

their, in a sense, "final" version of the axioms of classical logic, they noted: "Whether the system of axioms is complete at least in the sense that all the logical formulas which are correct for each domain of individuals can actually be derived from them, is still an unsolved question."

(quoted after
**S. C. Kleene**. The Work of Kurt Gödel. "The Journal of Symbolic Logic", December 1976, Vol.41, N4, pp.761-778
See also: Hilbert and Ackermann's 1928 Logic Book by Stanley N. Burris).

Indeed, as we will verify below, a) all axioms of the classical logic ($L_1$-$L_{11}$, $L_{12}$-$L_{15}$) are logically valid, b) the inference rules MP, Gen allow to prove

(from logically valid formulas) only logically valid formulas. Hence, in this way only logically valid formulas can be proved. Still, is our list of logical axioms complete in the sense that **all logically valid formulas can be proved?** − the question asked by Hilbert and Ackermann in 1928. The answer is "yes" − as Kurt Gödel established in 1929, in his doctoral dissertation "Ueber die Vollständigkeit des Logikkalküls"(visit Gödel's archive at the Princeton University Library). The corresponding paper appeared in 1930:

**K. Gödel.** Die Vollständigkeit der Axiome des logischen Funktionenkalküls. "Monatshefte fuer Mathematik und Physik", 1930, Vol.37, pp.349-360.

**Theorem 4.3.1 (Gödel's Completeness Theorem).** In any predicate language, a formula is logically valid, if and only if it can be proved by using the classical logic [$L_1$-$L_{11}$, $L_{12}$-$L_{15}$, MP, Gen].

In fact, a more general theorem can be proved:

**Theorem 4.3.2** (Thanks to Sune Foldager for the idea.). If T is a first order theory with classical logic, then some formula F is true in all models of T, if and only if T proves F.

Gödel's Completeness Theorem follows from Theorem 4.3.2 (simply consider a theory T with an empty set of specific axioms).

First, let us prove the easy part (the **soundness theorem**) − that all the formulas that can be proved by using the classical logic [$L_1$-$L_{11}$, $L_{12}$-$L_{15}$, MP, Gen] are logically valid.

**Lemma 4.3.1.** All the axioms of the classical logic ($L_1$-$L_{11}$, $L_{12}$-$L_{15}$) are logically valid.

**Proof.**

1) Under the classical truth tables, the propositional axioms $L_1$-$L_{11}$ take only true values (Lemma 4.2.1). Hence, these axioms are true under all interpretations.

2a) **$L_{12}$**: $\forall xF(x) \rightarrow F(t)$, where F is any formula, and t is a term such that the substitution $F(x/t)$ is admissible.

Let us assume that, under some interpretation J, for some values of its free variables, $L_{12}$ is false. According to the classical truth tables, this could be only, if and only if $\forall xF(x)$ were true, and $F(t)$ were false (under the interpretation J, for the same above-mentioned values of free variables). Let us "compute" the value of the term t for these values of free variables (since the substitution $F(x/t)$ is admissible, t may contain only these variables), and denote it by c. Thus, $F(c)$ is false. But $\forall xF(x)$ is true, hence, $F(a)$ is true for all $a \in D_J$ , i.e. $F(c)$ also is true. Contradiction. Hence, $L_{12}$ is true under all

interpretations for all combinations of its free variables.

2b) **L$_{13}$**: F(t)→∃xF(x), where F is any formula, and t is a term such that the substitution F(x/t) is admissible.

Similarly, see Exercise 4.3.1.

2c) **L$_{14}$**: ∀x(G→F(x))→(G→∀xF(x)), where F is any formula, and G is a formula that does not contain x as a free variable.

Let us assume that, under some interpretation J, for some values of its free variables, L$_{14}$ is false. According to the classical truth tables, this could be only, if and only if ∀x(G→F(x)) were true, and G→∀xF(x) were false (under the interpretation J, for the same above-mentioned values of free variables)

If ∀x(G→F(x)) is true, then G→F(c) is true for all $c \in D_J$ . Since G does not contain x, this means that if G is true, then F(c) is true for all $c \in D_J$ .

From the orher side, if G→∀xF(x) is false, then G is true, and ∀xF(x) is false. And finally, if ∀xF(x) is false, then F(c) is false for some $c \in D_J$ . But, as we established above, if G is true, then F(c) is true for all $c \in D_J$ . Contradiction. Hence, under all interpretations, L$_{14}$ is true for all combinations of its free variables.

2d) **L$_{15}$**: ∀x(F(x)→G)→(∃xF(x)→G), where F is any formula, and G is a formula that does not contain x as a free variable.

Similarly, see Exercise 4.3.1.

Q.E.D.

**Exercise 4.3.1.** Verify that the axioms L$_{13}$ and L$_{15}$ are logically valid as well.

**Lemma 4.3.2.** From logically valid formulas, the inference rules MP and Gen allow deriving only of logically valid formulas..

**Proof.**

1. **Modus Ponens**. Assume, B and B→C are logically valid formulas. By MP, we derive C. Assume, C is **not** logically valid, i.e., under some interpretation J, for some values of its free variables, C is false. Under this interpretation J, for these values of free variables of C, the formulas B and B→C are true. Then, according to the classical truth tables, C also must be true. Contradiction. Hence, C is logically valid.

2. **Generalization**. Assume, F(x) is logically valid, but ∀xF(x) is not, i.e., under some interpretation J, for some values of its free variables, ∀xF(x) is false. Hence, under this interpretation J, for these values of free variables of ∀xF(x), there is $c \in D_J$ such that F(c) is false. But F(x) is logically valid, i.e.

F(c) is true. Contradiction. Hence, $\forall xF(x)$ is logically valid.

Q.E.D.

**Theorem 4.3.3 (soundness of the classical predicate logic)**. All the formulas that can be proved by using the classical logic [$L_1$-$L_{11}$, $L_{12}$-$L_{15}$, MP, Gen], are logically valid.

**Proof.** Immediately, by Lemmas 4.3.1 and 4.3.2.

**Exercise 4.3.2.** Verify that if, under an interpretation J, all specific **axioms** of a theory T are true, then all **theorems** of T also are true under J. (Hint: each theorem C is proved by using some finite set of specific axioms, let us denote by B the conjunction of these axioms, consider the formula B→C, and use Theorem 4.3.3.)

Of course, the above soundness theorem is the easy half of Gödel's Completeness Theorem. To complete the proof, we must prove the converse: if some formula is logically valid, then it can be proved by using the classical logic [$L_1$-$L_{11}$, $L_{12}$-$L_{15}$, MP, Gen].

## Model Existence Theorem

Gödel's Completeness Theorem (and Theorem 4.3.2) are easy consequences of the so-called Model Existence Theorem (see below), first proved in the above-mentioned Gödel's paper of 1930.

Gödel's initial proof was simplified in 1947, when Leon Henkin presented in his Ph.D. thesis a new proof of the so-called Model Existence Theorem (see below). The result was published in 1949:

**L. Henkin.** The completeness of the first-order functional calculus. "J. Symbolic Logic", 1949, vol.14, pp.159-166.

See also Henkin's later account of his discovery:

**L. Henkin.** The discovery of my completeness proofs. "The Bulletin of Symbolic Logic", 1996, vol.2, N2, pp.127-158.

An even simpler version Henkin's proof was found independently and almost simultaneously by Gisbert Hasenjäger, however, when publishing, he acknowledged Henkin's priority:

**G. Hasenjäger**. Eine Bemerkung zu Henkin's Beweis für die Vollständigkeit des Prädikatenkalküls der ersten Stufe. "J. Symbolic Logic", 1953, vol.18, pp.42-48.

If T is an inconsistent theory, then there are no models of T. Indeed, if T proves a contradiction, i.e. a formula of the kind $B \wedge \neg B$, then, in a model of T, the formula B must be true and false simultaneously. This is impossible.

Hence, if there is a model of T, then T is consistent.

The converse question: could it be possible that T is a consistent theory, but there are no models of T? The answer is given in the

**Theorem 4.3.4 (Model Existence Theorem).** If a first order classical formal theory is consistent (in the sense that, by using the classical logic, it does not prove contradictions), then there is a finite or countable model of this theory (i.e. an interpretation with a finite or countable domain, under which all axioms and theorems of the theory are true).

In the 1920s, some people insisted that mere consistency of a theory (in the syntactic sense of the word − as the lack of contradictions) is not sufficient to regard it as a meaningful theory − as a "theory of something". Model Existence Theorem says the contrary − (syntactic!) consistency of a theory **is** sufficient: **if a theory does not contain contradictions, then it is a "theory of something"** − it describes at least some kind of "mathematical reality". For example, you may think that Euclidean geometry is "meaningless" − because it does not describe 100% correctly the spacial properties of the Universe. But it's your problem, not Euclid's − use another theory, if necessary. Euclidean geometry describes its own kind of "mathematical reality" – and 100% correctly!

Let us assume the Model Existence Theorem (we will prove it later in this Section).

**Proof of Theorem 4.3.2.**

If T proves F, then F is true in all models of T (Exercise 4.3.2).

Now, let us assume that some formula F is true in all models of theory T, yet it cannot be proved in T. Let us consider the theory T' in the language of T which contains (besides the axioms of T) an additional non-logical axiom − the negation of F, i.e. the formula $\neg \forall x_1 ... \forall x_n F$, where $x_1, .., x_n$ are exactly all the free variables of F (if F contains free variables $x_1, .., x_n$, then, to negate its assertion, we must add to F the quantifiers $\forall x_1 ... \forall x_n$). Since F cannot be derived from the axioms of T, **T' is a consistent theory.**

Indeed, if T' would be inconsistent, i.e. we could prove in T' some formula C and its negation $\neg$C, then we had proofs of [T]: $\neg \forall x_1 ... \forall x_n F \vdash C$, and [T]: $\neg \forall x_1 ... \forall x_n F \vdash \neg C$. Since $\neg \forall x_1 ... \forall x_n F$ is a closed formula, by Deduction Theorem 2, [T]: $\vdash \neg \forall x_1 ... \forall x_n F \rightarrow C$, and [T]: $\vdash \neg \forall x_1 ... \forall x_n F \rightarrow \neg C$. Now, by axiom $L_9$: $(B \rightarrow C) \rightarrow (B \rightarrow \neg C) \rightarrow \neg B$, we obtain that [T]: $\vdash \neg\neg \forall x_1 ... \forall x_n F$. By the (classical) Double Negation Law, this implies [T]: $\vdash \forall x_1 ... \forall x_n F$, and by axiom $L_{12}$: $\forall x B(x) \rightarrow B(x)$ − [T]: $\vdash F$. But, by our assumption, F cannot be

proved in T. Hence, T' is a consistent theory.

Now, by the Model Existence Theorem, there is a model of T', i.e. an interpretation J that makes all its axioms true. Under this interpretation, all axioms of T are true, i.e. J is a model of T. And the formula $\neg\forall x_1...\forall x_n F$ (as an axiom of T') also is true under J. On the other hand, since F is true in all models of T, it is true also under the interpretation J. Hence, formulas $\forall x_1...\forall x_n F$ and $\neg\forall x_1...\forall x_n F$ both are true under J. This is impossible, hence, F must be provable in T. Q.E.D.

1. Such a simple proof seems almost impossible! We are proving that the logical axioms and rules of inference are strong enough, but where come these axioms in? They come in − in the proof of the Model Existence Theorem. This theorem says that if some formal theory T does not have models, then the logical axioms and rules of inference are strong enough to derive a contradiction from the axioms of T. But the proof of the Model Existence Theorem that we will consider below, is positive, not negative!

2. The above simple proof seems to be extremely non-constructive! "If F is true in all models of T, then it can be proved in T". How could we obtain this proof? Still, how do we know that F is true in all models of T? Only, if we had a constructive procedure that is verifying this, we could ask for an algorithm converting such procedures into proofs in T!

**Proof of the Model Existence Theorem**

**Exercise 4.3.3** (optional, for smart students). Prove the Model Existence Theorem by using the following smart ideas due to L. Henkin and G. Hasenjäger. Let T be a consistent theory. We must build a model of T. What kind of "bricks" should we use for this "building"? **Idea #1:** let us use object constants of the language! So, let us add to the language of T an infinite set of new object constants $d_1$, $d_2$, $d_3$, ... (and adopt the corresponding additional instances of logical axioms). Prove that this extended theory $T_0$ is consistent. The model we are building must contain all "objects" whose existence can be proved in $T_0$. **Idea #2:** for each formula F of $T_0$ having exactly one free variable (for example, x) let us add to the theory $T_0$ the axiom $\exists x F(x) \to F(d_i)$, where the constant $d_i$ is unique for each F. If $T_0$ proves $\exists x F(x)$, then this constant $d_i$ will represent in our model the "object" x having the property F. Prove that this extended theory $T_1$ is consistent. Idea #3: prove the (non-constructive) Lindenbaum's lemma: the axiom set of any consistent theory can be extended in such a way, that the extended theory is consistent and complete (the axiom set of the extended theory may be not algorithmically solvable).

After this, extend $T_1$ to a consistent complete theory $T_2$. **Idea #4:** let us take as the domain of the interpretation M the set of all those terms of $T_0$ that do not contain variables. And let us interpret each function constant f as the "syntactic constructor function" f', i.e. let us define the value $f'(t_1, ..., t_n)$ simply as the character string "$f(t_1, ..., t_n)$". Finally, let us interpret each predicate constant p as the relation p' such that $p'(t_1, ..., t_n)$ is true in M, if and only if $T_2$ proves $p'(t_1, ..., t_n)$. To complete the proof, prove that an arbitrary formula G is true in M, if and only if $T_2$ proves G. Hence, all theorems of the initial theory T are true in M.

Adolf Lindenbaum (1904-1941), his wife Janina Hosiasson-Lindenbaum (1899-1942).

**Lemma 4.3.4 (Lindenbaum's Lemma)**. Any consistent first order theory can be extended to a consistent complete theory. More precisely, if T is a consistent first order theory, then, in the language of T, there is a set A of closed formulas such that T+A is a consistent complete theory. (In general, T+A is not a formal theory in the sense of Section 1.1, see below.)

**Note**. By T+A we denote the first order theory in the language of T, obtained from T by adding the formulas of the set A as non-logical axioms.

**Exercise 4.3.4.** Verify that, in any predicate language L, only countably many formulas can be generated. I.e. produce an algorithm for printing out a sequence $F_0$, $F_1$, $F_2$, ... containing all the formulas of L.

**Proof of Lindenbaum's Lemma (Attention: non-constructive reasoning!)**

Let us use the algorithm of the Exercise 4.3.4 printing out the sequence $F_0$, $F_1$, $F_2$, ... of all formulas in the language of T, and let us run through this sequence, processing only those formulas $F_i$ that are closed.

At the very beginning, the set of new axioms $A_0$ is empty.

At the step i, we already have some set $A_{i-1}$ of new axioms. If the formula $F_i$ is not closed, let us ignore it, and set $A_i = A_{i-1}$. Now, let us suppose that $F_i$ is a closed formula. If $T+A_{i-1}$ proves $F_i$, or $T+A_{i-1}$ proves $\neg F_i$, then we can ignore this formula, and set $A_i = A_{i-1}$. If T+A does not prove neither $F_i$, nor $\neg F_i$, then let us simply add $F_i$ (or $\neg F_i$, if you like it better) to our set of new axioms, i.e. set $A_i = A_{i-1} \cup \{Fi\}$.

Etc., *ad infinitum*. As the result of this process we obtain a set of closed formulas $A = A_0 \cup A_1 \cup A_2 \cup ... \cup A_i \cup ...$.

Let us prove that T+A is a consistent complete theory.

**Consistency.** If T+A would be inconsistent, we would have a proof of [T+A] ⊢ $C \land \neg C$ for some formula C. If, in this proof, no axioms from the set A would be used, we would have a proof of [T] ⊢ $C \land \neg C$ , i.e. T would be inconsistent.

Otherwise, the proof of [T+A] ⊢ $C \land \neg C$ could contain a finite number of axioms $B_1$, ..., $B_k$ from the set A. Let us arrange these axioms in the sequence, as we added them to the set A. Thus we have a proof of [T]: $B_1$, ..., $B_k$ ⊢ $C \land \neg C$ . Let us remind Theorem 2.4.1(a):

If $A_1$, $A_2$, ..., $A_n$, B ⊢ $C \land \neg C$ , then $A_1$, $A_2$, ..., $A_n$ ⊢ ¬B.

Hence, we have a proof of [T]: $B_1$, ..., $B_{k-1}$ ⊢ ¬$B_k$. But this is impossible − we added $B_k$ to the set A just because T+$A_{i-1}$ could not prove neither $B_k$, nor ¬$B_k$. Q.E.D.

**Completeness.** We must verify that, for any closed formula F in the language of T, either T+A ⊢ F, or T+A ⊢ ¬F. Let us assume, this is not the case for some closed formula F. Of course, F appears in the above sequence $F_0$, $F_1$, $F_2$, ... as some $F_i$. If neither T+A ⊢ F, nor T+A ⊢ ¬F, then neither T+$A_{i-1}$ ⊢ $F_i$, nor T+$A_{i-1}$ ⊢ ¬$F_i$. In such a situation we would add F to the set A, hence, we would have T+A ⊢ F. Q.E.D.

This completes the proof of Lindenbaum's Lemma.

**Attention: non-constructive reasoning!** T+A is a somewhat strange theory, because, in general, we do not have an algorithmic decision procedure for its axiom set. Indeed, to decide, is some closed formula F an axiom of T+A, or not, we must identify F in the sequence $F_0$, $F_1$, $F_2$, ... as some $F_i$, and after this, we must verify, whether T+$A_{i-1}$ proves $F_i$, or T+$A_{i-1}$ proves ¬$F_i$, or none of these. Thus, in general, T+A is not a formal theory in the sense of [Section 1.1](#).

**Proof of the Model Existence Theorem (Attention: non-constructive reasoning!)**

Inspired by the beautiful exposition in [Mendelson [1997]](#).

**Step 1.** We must build a model of T. What kind of "bricks" should we use for this "building"? Idea #1: let us use object constants of the language! So, in order to prepare enough "bricks", let us add to the language of T a countable set of new object constants $d_1$, $d_2$, $d_3$, ... (and extend the definitions of terms, atomic formulas and formulas accordingly, and add new instances of logical axioms accordingly). Let us prove that, if T is consistent, then this extended theory $T_0$ also is consistent.

If $T_0$ would be inconsistent, then, for some formula C, we could obtain a proof of $[T_0]$: $\vdash C \land \neg C$. If, in this proof, object constants from the set {$d_1$, $d_2$, $d_3$, ...} would not appear at all, then, in fact, we had a proof of [T]: $\vdash C \land \neg C$, i.e. we could conclude that T is inconsistent. But, if the new object constants do appear in the proof of $[T_0]$: $\vdash C \land \neg C$ ? Then, let us replace these constants by any variables of T that do not appear in this proof (this is possible, since each predicate language contains a countable set of object variables). After these substitutions, the proof becomes a valid proofs of T, because:

a) The logical axioms remain valid.

b) The non-logical axioms of T do not contain the object constants $d_1$, $d_2$, $d_3$, ..., i.e. they do not change.

c) Applications of inference rules MP and Gen remain valid.

Hence, [T]: $\vdash C' \land \neg C'$, where the formula C' has been obtained from C by the above substitutions. I.e., if $T_0$ would be inconsistent, then T also would be inconsistent.

**Step 2.** The model we are building must contain all "objects" whose existence can be proved in $T_0$. Idea #2: for each formula F of $T_0$ having exactly one free variable (for example, x) let us add to the theory $T_0$ the axiom $\exists x F(x) \rightarrow F(d_i)$, where the constant $d_i$ is unique for each F. If $T_0$ proves $\exists x F(x)$, then this $d_i$ will represent in our model the "object" x having the property F. Let us prove that, if T is consistent, then this extended theory $T_1$ also is consistent. Note that in $T_1$ the same language is used as in $T_0$.

To implement the Idea #2 correctly, first let us use the algorithm of the Exercise 4.3.4 printing out the sequence $F_0$, $F_1$, $F_2$, ... of all formulas in the language of $T_0$, and let us run through this sequence, processing only those formulas $F_i$ that have exactly one free variable. Let us assign to each such formula $F_i$ a unique constant $d_{c(i)}$ in such a way that $d_{c(i)}$ does not appear neither in the non-logical axioms of T, nor in $F_i$, nor in the axioms $\exists y F_j(y) \rightarrow F_j(d_{c(j)})$ for all formulas $F_j$ preceding $F_i$ in the sequence $F_0$, $F_1$, $F_2$, .... And, if x is the (only) free variable of $F_i$, let us adopt $\exists x F_i(x) \rightarrow F_j(d_{c(i)})$ as an axiom of $T_1$.

Now, let us assume that the extended theory $T_1$ is inconsistent, i.e. that, for some formula C of $T_0$, we have a proof of $[T_1]$: $\vdash C \land \neg C$. In these proofs,

only a finite number n of axioms $\exists xF(x) \rightarrow F(d_{c(F)})$ could be used. If n=0, then we have $[T_0]$: $\vdash C \wedge \neg C$ , i.e. then $T_0$ is inconsistent.

If n>0, then let us mark the axiom $\exists xF(x) \rightarrow F(d_{c(F)})$ with F having the largest index in the sequence $F_0$, $F_1$, $F_2$, .... And, in the proof of $[T_1]$: $\vdash C \wedge \neg C$ , let us replace the constant c(F) by some variable y that does not appear in this proof (this is possible, since each predicate language contains a countable set of variables). After this substitution, the proof remain a valid proof of $T_1$, because:

a) The logical axioms remain valid.

b) The non-logical axioms of T do not contain the constant c(F), i.e. they do not change.

c) The axiom $\exists xF(x) \rightarrow F(d_{c(F)})$ becomes $\exists xF(x) \rightarrow F(y)$. Since F does not contain the constant c(F), the premise $\exists xF(x)$ does not change.

d) The remaining n-1 axioms $\exists yF_j(y) \rightarrow F_j(d_{c(j)})$ of $T_1$ do not contain the constant c(F), i.e. they do not change.

e) Applications of inference rules MP and Gen remain valid.

Thus we have now another proof of a contradiction − $[T_1]$: $\vdash C' \wedge \neg C'$ , where the formula C' has been obtained from C by substituting y for c(F). Let us remind Theorem 2.4.1(a):

$\qquad$ If $A_1$, $A_2$, ..., $A_n$, B $\vdash C \wedge \neg C$ , then $A_1$, $A_2$, ..., $A_n$ $\vdash \neg B$.

Let us take the formula $\exists xF(x) \rightarrow F(y)$ for B, and C'- for C. Thus, there is a proof of $\neg(\exists xF(x) \rightarrow F(y))$, where only logical axioms, non-logical axioms of T, and the remaining n-1 axioms $\exists yF_j(y) \rightarrow F_j(d_{c(j)})$ of $T_1$ are used. Let us remind the Exercise 2.6.3(b) $[L_1\text{-}L_{11}$, MP]: $\vdash \neg(A \rightarrow B) \leftrightarrow A \wedge \neg B$ . Thus, $\neg(\exists xF(x) \rightarrow F(y))$ is equivalent to $\exists xF(x) \wedge \neg F(y)$ , and, in fact, we have a proof of $\exists xF(x)$, and a proof of $\neg F(y)$. By applying Gen to the second formula, we obtain a proof of $\forall y \neg F(y)$, which is equivalent to $\neg \exists yF(y)$ (indeed, let us remind Section 3.2, Table 3.2, Group IV, constructively, $\vdash \forall x \neg B \leftrightarrow \neg \exists xB$). By Replacement Theorem 3, $\neg \exists yF(y)$ is equivalent to $\neg \exists xF(x)$. Thus, we have a proof of a contradiction $\exists xF(x) \wedge \neg \exists xF(x)$ , where only logical axioms, non-logical axioms of T, and the remaining n-1 axioms $\exists yF_j(y) \rightarrow F_j(d_{c(j)})$ of $T_1$ are used.

Let us repeat the above chain of reasoning another n-1 times to eliminate **all** occurrences of the axioms $\exists xF(x) \rightarrow F(d_{c(F)})$ from our proof of a contradiction. In this way we obtain a proof of a contradiction in $T_0$, which is impossible (see

Step 1). Hence, $T_1$ is a consistent theory.

**Step 3.** Idea #3: let us use the (non-constructive!) Lindenbaum's lemma, and extend $T_1$ to a consistent complete theory $T_2$. Note that in $T_2$ the same language is used as in $T_0$.

**Step 4.** Let us define an interpretation M of the language of $T_0$, in which all theorems of $T_2$ will be true. Since all theorems of the initial theory T are theorems of $T_2$, this will complete our proof.

Idea #4: let us take as the domain $D_M$ of the interpretation M the (countable! − verify!) set of all constant terms of $T_0$, i.e. terms that do not contain variables (this set of terms is not empty, it contains at least the countable set of object constants added in Step 1). And let us define interpretations of object constants, function constants and predicate constants as follows.

a) The interpretation of each object constant c is the constant c itself.

b) The interpretation of a function constant f is the "syntactic constructor function" f', i.e., if f is an n-ary function constant, and $t_1$, ..., $t_n$ are constant terms, then the value $f'(t_1, ..., t_n)$ is defined simply as the character string "$f(t_1, ..., t_n)$" (quotation marks ignored).

c) The interpretation of a predicate constant p is the relation p' such, if p is an n-ary predicate constant, and $t_1$, ..., $t_n$ are constant terms, then $p'(t_1, ..., t_n)$ is defined as true in M, if and only if $T_2$ proves $p(t_1, ..., t_n)$ (note that $T_2$ is a consistent complete theory, i.e. it proves either $p(t_1, ..., t_n)$, or $\neg p(t_1, ..., t_n)$, but not both!).

**Step 5**. To complete the proof, we must prove that, in the language of $T_0$, an arbitrary formula G is true in M, if and only if $T_2$ proves G (let us denote this, as usual, by $T_2 \vdash G$. This will be proved, if we will prove that, if $x_1$, ..., $x_m$ is the set of at least all free variables contained in the formula G, and $t_1$, ..., $t_m$ are constant terms, then $G(t_1, ..., t_m)$ is true in M, if and only if $T_2 \vdash G(t_1, ..., t_m)$. The proof will be by induction.

**Induction base**: G is an atomic formula $p(s_1, ..., s_n)$, where p is a predicate constant, and the terms $s_1$, ..., $s_n$ contain some of the variables $x_1$, ..., $x_m$. In $s_1$, ..., $s_n$, let us substitute for $x_1$, ..., $x_m$ the terms $t_1$, ..., $t_m$ respectively. In this way we obtain constant terms $s'_1$, ..., $s'_n$. Thus $G(t_1, ..., t_m)$ is simply $p(s'_1, ..., s'_n)$. By definition (see Step 4), $p(s'_1, ..., s'_n)$ is true, if and only if $T_2 \vdash p(s'_1, ...,$

s'$_n$), i.e., if and only if $T_2 \vdash G(t_1, ..., t_m)$. Q.E.D.

**Induction step.**

**Note.** Since, $T_2$ is a complete consistent theory, for any closed formula F, $T_2$ proves either F, or ¬F (but not both). Hence, if we know that F is true in M, if and only if $T_2 \vdash F$, then we can conclude that F is false in M, if and only if $T_2 \vdash ¬F$. Indeed, if F is false, then F is not true, i.e. $T_2$ does not prove F, i.e. $T_2 \vdash ¬F$. And, if $T_2 \vdash ¬F$, then $T_2$ does not prove F, i.e. F is not true, i.e. F is false. And conversely: if we know that F is false in M, if and only if $T_2 \vdash ¬F$, then we can conclude that F is true in M, if and only if $T_2 \vdash F$. Indeed, if F is true, then ¬F is not true, i.e. $T_2$ does not prove ¬F, i.e. $T_2 \vdash F$. And, if $T_2 \vdash F$, then $T_2$ does not prove ¬F, i.e. F is not false, i.e. F is true.

**Case 1:** G is ¬H. Then, according to the classical truth tables, $G(t_1, ..., t_m)$ is true in M, if and only if $H(t_1, ..., t_m)$ is false in M. By the induction assumption, $H(t_1, ..., t_m)$ is true in M, if and only if $T_2 \vdash H(t_1, ..., t_m)$. Then, by the above note, since $H(t_1, ..., t_m)$ is a closed formula, $H(t_1, ..., t_m)$ is false in M, if and only if $T_2 \vdash ¬H(t_1, ..., t_m)$, i.e., if and only if $T_2 \vdash G(t_1, ..., t_m)$. Q.E.D.

**Case 2:** G is H→K. Then, according to the classical truth tables, $G(t_1, ..., t_m)$ is false in M, if and only if $H(t_1, ..., t_m)$ is true in M, and $K(t_1, ..., t_m)$ is false in M. By the induction assumption, $H(t_1, ..., t_m)$ is true in M, if and only if $T_2 \vdash H(t_1, ..., t_m)$, and $K(t_1, ..., t_m)$ is true in M, if and only if $T_2 \vdash K(t_1, ..., t_m)$. By the above note, $K(t_1, ..., t_m)$ is false in M, if and only if $T_2 \vdash ¬K(t_1, ..., t_m)$. Hence,

$G(t_1, ..., t_m)$ is false in M, if and only if

$T_2 \vdash H(t_1, ..., t_m)$, and $T_2 \vdash ¬K(t_1, ..., t_m)$,

or,

$G(t_1, ..., t_m)$ is true in M, if and only if

not ($T_2 \vdash H(t_1, ..., t_m)$, and $T_2 \vdash ¬K(t_1, ..., t_m)$).

Let us remind Theorem 2.2.1 and Exercise 2.6.3(a), [$L_1$-$L_{11}$, MP]: $\vdash (A \to B) \leftrightarrow ¬(A \land ¬B)$ . In $T_2$, all the axioms of the classical logic are adopted, hence (verify!),

$G(t_1, ..., t_m)$ is true in M, if and only if $T_2 \vdash H(t_1, ..., t_m) \rightarrow K(t_1, ..., t_m)$,

or,

$\qquad G(t_1, ..., t_m)$ is true in M, if and only if $T_2 \vdash G(t_1, ..., t_m)$.

Q.E.D.

**Case 3:** G is $H \wedge K$ . Then, according to the classical truth tables, $G(t_1, ..., t_m)$ is true in M, if and only if $H(t_1, ..., t_m)$ is true in M, and $K(t_1, ..., t_m)$ is true in M. By the induction assumption, $H(t_1, ..., t_m)$ is true in M, if and only if $T_2 \vdash H(t_1, ..., t_m)$, and $K(t_1, ..., t_m)$ is true in M, if and only if $T_2 \vdash K(t_1, ..., t_m)$. Let us remind Theorem 2.2.1. In $T_2$, all the axioms of the classical logic are adopted, hence (verify!),

$\qquad G(t_1, ..., t_m)$ is true in M, if and only if $T_2 \vdash H(t_1, ..., t_m) \wedge K(t_1, ..., t_m)$ ,

or,

$\qquad G(t_1, ..., t_m)$ is true in M, if and only if $T_2 \vdash G(t_1, ..., t_m)$.

Q.E.D.

**Case 4:** G is $H \vee K$ . Then, according to the classical truth tables, $G(t_1, ..., t_m)$ is false in M, if and only if $H(t_1, ..., t_m)$ is false in M, and $K(t_1, ..., t_m)$ is false in M. By the induction assumption, and by the above note, $H(t_1, ..., t_m)$ is false in M, if and only if $T_2 \vdash \neg H(t_1, ..., t_m)$, and $K(t_1, ..., t_m)$ is false in M, if and only if $T_2 \vdash \neg K(t_1, ..., t_m)$. Let us remind Theorem 2.2.1 and Theorem 2.4.10(b): $[L_1\text{-}L_{10}, MP] \vdash \neg(A \vee B) \leftrightarrow \neg A \wedge \neg B$ (the so-called Second de Morgan Law). In $T_2$, all the axioms of the classical logic are adopted, hence (verify!),

$G(t_1, ..., t_m)$ is false in M, if and only if $T_2 \vdash \neg(H(t_1, ..., t_m) \vee K(t_1, ..., t_m))$ ,

or, $G(t_1, ..., t_m)$ is false in M, if and only if $T_2 \vdash \neg G(t_1, ..., t_m)$. Thus, by the above note, $G(t_1, ..., t_m)$ is true in M, if and only if $T_2 \vdash G(t_1, ..., t_m)$. Q.E.D.

**Case 5:** G is $\exists x H$. Then, by definition, $G(t_1, ..., t_m)$ is true in M, if and only if $H(x, t_1, ..., t_m)$ is "true for some x", i.e., if and only if $H(t, t_1, ..., t_m)$ is true in M for some constant term t in M. By the induction assumption, $H(t, t_1, ..., t_m)$ is true in M, if and only if $T_2 \vdash H(t, t_1, ..., t_m)$. Let us remind our above Step 2. Since $H(x, t_1, ..., t_m)$ is a formula containing exactly one free variable, in $T_2$ we have an axiom $\exists x H(x, t_1, ..., t_m) \rightarrow H(c_H, t_1, ..., t_m)$, where $c_H$ is an object

constant.

First, let us assume that $G(t_1, ..., t_m)$ is true in M. Then $H(t, t_1, ..., t_m)$ is true in M for some constant term t in M, hence, $T_2 \vdash H(t, t_1, ..., t_m)$ for this particular t. Let us remind the axiom $L_{13}$: $F(t) \rightarrow \exists x F(x)$. Since t is a constant term, this axiom is valid for t. We need the following instance of $L_{13}$: $H(t, t_1, ..., t_m) \rightarrow \exists x H(x, t_1, ..., t_m)$. In $T_2$, all the axioms of the classical logic are adopted, hence, $T_2 \vdash H(t, t_1, ..., t_m) \rightarrow \exists x H(x, t_1, ..., t_m)$, and, by MP, $T_2 \vdash \exists x H(x, t_1, ..., t_m)$, i.e. $T_2 \vdash G(t_1, ..., t_m)$. Q.E.D.

Now, let us assume that $T_2 \vdash G(t_1, ..., t_m)$, i.e. $T_2 \vdash \exists x H(x, t_1, ..., t_m)$. By the above-mentioned axiom, $T_2 \vdash \exists x H(x, t_1, ..., t_m) \rightarrow H(c_H, t_1, ..., t_m)$, where $c_H$ is an object constant. Thus, by MP, $T_2 \vdash H(c_H, t_1, ..., t_m)$. Since $c_H$ is a constant term, by the induction assumption, if $T_2 \vdash H(c_H, t_1, ..., t_m)$, then $H(c_H, t_1, ..., t_m)$ is true in M. Hence, $H(c_H, t_1, ..., t_m)$ is true in M, i.e. $H(x, t_1, ..., t_m)$ is true "for some x", i.e. $\exists x H(x, t_1, ..., t_m)$ is true in M, i.e. $G(t_1, ..., t_m)$ is true in M. Q.E.D.

**Case 6:** G is $\forall x H$. By definition, $G(t_1, ..., t_m)$ is true in M, if and only if $H(x, t_1, ..., t_m)$ is "true for all x", i.e., if and only if $H(t, t_1, ..., t_m)$ is true in M for all constant terms t in M. By the induction assumption, $H(t, t_1, ..., t_m)$ is true in M, if and only if $T_2 \vdash H(t, t_1, ..., t_m)$.

Let us prove that

$G(t_1, ..., t_m)$ is false in M, if and only if $T_2 \vdash \exists x \neg H(x, t_1, ..., t_m)$.

First, let us assume that $G(t_1, ..., t_m)$ is false in M. Then $H(t, t_1, ..., t_m)$ is false in M for some constant term t in M. By the induction assumption, and by the above note, $T_2 \vdash \neg H(t, t_1, ..., t_m)$. As in the Case 5, let us remind the axiom $L_{13}$: $\neg H(t, t_1, ..., t_m) \rightarrow \exists x \neg H(x, t_1, ..., t_m)$. In $T_2$, all the axioms of the classical logic are adopted, hence, by MP, $T_2 \vdash \exists x \neg H(x, t_1, ..., t_m)$.

Now, let us assume that $T_2 \vdash \exists x \neg H(x, t_1, ..., t_m)$. As in the Case 5, let us remind the axiom $\exists x \neg H(x, t_1, ..., t_m) \rightarrow \neg H(c_{\neg H}, t_1, ..., t_m)$, where $c_{\neg H}$ is an object constant. Hence, by MP, $T_2 \vdash \neg H(c_{\neg H}, t_1, ..., t_m)$, i.e. $T_2$ does not prove $H(c_{\neg H}, t_1, ..., t_m)$. Then, by the induction assumption, $H(c_{\neg H}, t_1, ..., t_m)$ is false in M, i.e. $\forall x H(x, t_1, ..., t_m)$ is false in M, i.e $G(t_1, ..., t_m)$ is false in M.

Thus, we know that $G(t_1, ..., t_m)$ is true in M, if and only if $T_2$ does not prove $\exists x \neg H(x, t_1, ..., t_m)$. Since $T_2$ is a complete theory, $G(t_1, ..., t_m)$ is true in M, if and only if $T_2 \vdash \neg \exists x \neg H(x, t_1, ..., t_m)$. Now, let us remind from Section 3.2, Table 3.2, Group I, $[L_1\text{-}L_{11}, L_{12}\text{-}L_{15}, MP, Gen]$: $\vdash \neg \exists x \neg B \leftrightarrow \forall x B$. In $T_2$, all the axioms of the classical logic are adopted, hence, $T_2 \vdash \neg \exists x \neg H(x, t_1, ..., t_m)$, if and only if $T_2 \vdash \forall x H(x, t_1, ..., t_m)$, i.e. $G(t_1, ..., t_m)$ is true in M, if and only if $T_2 \vdash G(t_1, ..., t_m)$. Q.E.D.

This completes the proof of the Model Existence Theorem. Q.E.D.

**Attention: non-constructive reasoning!** The above construction of the model M seems to be "almost constructive". The domain $D_M$ consists of all constant terms from the language of $T_0$. The axiom set of $T_1$ is algorithmically solvable (verify!). The interpretations of function constants are computable functions (verify!). But the interpretations of predicate constants? We interpreted each predicate constant p as the relation p' such that $p'(t_1, ..., t_n)$ is true, if and only if $T_2$ proves $p(t_1, ..., t_n)$. This relation would be, in general, not algorithmically solvable, even if the axiom set of $T_2$ would be solvable! But, in general, the axiom set of theory $T_2$ (obtained by means of Lindenbaum's Lemma) is not algorithmically solvable! Thus, our construction of the model M is essentially non-constructive.

**Exercise 4.3.5** (optional, course-work for smart students). Verify that the "degree of non-constructiveness" of the Model Existence Theorem is $\Delta_2^0$ in the so-called arithmetical hierarchy. This became possible due to the improvements introduced by G. Hasenjäger. Hint: verify that all the functions necessary for the proof are "computable in the limit". A function $f(x)$ is called computable in the limit, if and only if there is a computable function $g(x,y)$ such that, for all x, $f(x) = \lim_{y \to \infty} g(x, y)$ ).

**Exercise 4.3.6** (optional, course-work for smart students). If the language of first order theory T contains the equality predicate constant $x=y$, how this constant will be interpreted in the models, built for T according to the Model Existence Theorem? If the axioms of T imply the basic properties of equality (reflexivity, symmetry, transitivity and "indiscernibility if identicals") then only the following is guaranteed for the interpretation of $x=y$: it will always be interpreted as some equivalence relation over the domain of interpretation. In some contexts, so-called **normal models** are preferable, where equality interpreted as **equality of domain elements**. Re-prove Model Existence Theorem by using normal models only. Or, see Mendelson [1997].

## Consequences of Gödel's Completeness Theorem

From now on, we could forget our great ability of proving formulas in the classical predicate logic, that we learned in Section 3. Indeed, in order to

verify, is a formula provable in [L$_1$-L$_{15}$, MP, Gen], or not, we can, instead of trying a proof from axioms, try to verify is this formula is logically valid, or not. If it is, then, by Completeness Theorem, formula is provable, if not – it is not provable. For simple formulas, this method is really simpler than direct proving of formulas in [L$_1$-L$_{15}$, MP, Gen]. However, we must remember that for the general case, this method does not work (see Church-Kalmar Theorem below).

**Notion of logical consequence**

As noted above (Exercise 4.1.6), some formula G is a "logical consequence" of the formulas $A_1, \ldots, A_n$, if and only if the formula $A_1 \wedge \ldots \wedge A_n \rightarrow G$ is logically valid, hence, by Gödel's Completeness Theorem – if and only if, $A_1 \wedge \ldots \wedge A_n \rightarrow G$ is provable in the classical predicate logic [L$_1$-L$_{15}$, MP, Gen], or, equivalently, if

**G can be derived from A$_1$, ..., A$_n$ by using the axioms and rules of inference of the classical predicate logic.**

This completes the formalization of the initially somewhat mystical notion of "logical consequence", and allows to consider reasoning as a process that could be performed by using computers (see below).

**Consistency and satisfiability**

A set of formulas F$_1$, ..., F$_n$ is called **inconsistent**, if and only if a contradiction (i.e. a formula $B \wedge \neg B$) can be derived from it. For example, the set {B, B→C, C→¬B} is inconsistent (verify).

The Model Existence Theorem allows to connect the notions of consistency and satisfiability.

**Exercise 4.3.7.** Verify, that a set of formulas in a predicate language: a) is consistent in the classical logic, if and only if it is satisfiable, b) is inconsistent in the classical logic, if and only if it is unsatisfiable. (Hint: use the result of Exercise 4.1.1).

**Computational complexity of the problem**

**Corollary 4.3.1.** In any **predicate** language the set of all logically valid formulas is algorithmically enumerable. I.e. given a language L, we can write an algorithm that (working *ad infinitum*) prints out all the logically valid formulas of L (and only these formulas).

**Proof.** Immediately from Exercise 1.1.4 and Gödel's Completeness Theorem.

This makes Gödel's Completeness Theorem very significant: it shows that the "doubly non-constructive" notion of logically valid formula is at least 50% constructive – **semi-constructive**! Semi-feasible for computers!

Still, unfortunately, this notion appears to be not 100% constructive. In 1936, Alonzo Church proved that at least some predicate languages do not allow an algorithm determining, is a given formula logically valid or not (i.e. an algorithm solving the famous *Entscheidungsproblem* – the decision problem):

**A. Church.** A note on the Entscheidungsproblem. "Journal of Symb. Logic", 1936, vol.1, pp.40-41

After this, Laszlo Kalmar in

**L. Kalmar.** Die Zurückführung des Entscheidungsproblems auf den Fall von Formeln mit einer einzigen, binären Funktionsvariablen. "Compositio Math.", 1936, Vol.4, pp.137-144

improved Church's result:

**Church-Kalmar Theorem.** If a predicate language contains **at least one predicate constant that is at least binary**, then this language does not allow an algorithm determining, is a given formula of this language logically valid or not.

Thus, none of serious predicate languages allows such an algorithm (languages of PA and ZF included). For details, see Mendelson [1997].

Sometimes, this fact (the 50% constructiveness of the notion of the logical validity) is expressed a follows: the logical validity of predicate formulas is **semi-decidable**.

**Corollary**. If a predicate language contains at least one predicate constant that is at least binary, then this language does not allow an algorithm determining, does some formula G of this language follow from some other formulas $A_1$, ..., $A_n$. In other words – the **task of reasoning in such a language is not algorithmically solvable.**

**Exercise 4.3.8.** Verify, this.

**Note.** The proof method of Church-Kalmar Theorem is now a standard: try "embedding" the Halting Problem of Turing machines into the problem of logical validity. Namely, take a Turing machine M and build a (sophisticated) formula F, that is logically valid, if and only if M halts on empty input. For details, see Mendelson [1997].

**Church-Kalmar Theorem and knowledge bases**

If we will build our knowledge base by using some predicate language, then,

in general, the situation will be as follows:

a) We will have some set of constants registered in the knowledge base: **object constants**: $c_1$, $c_2$, ..., $c_k$, **function constants** and **predicate constants**: $p_1$, $p_2$, ..., $p_m$ (with argument numbers specified).

[The **Closed World Assumption**: in the world, there exist only objects denoted by our constants $c_1$, $c_2$, ..., $c_k$. In fact, this assumption should be represented as an axiom $\forall x$ ($x=c_1$ v $x=c_2$ v ... v $x=c_k$). The **Open World Assumption**: in the world, there exist more objects than denoted by our object constants.]

b) **Facts, concepts** and **rules** ("laws") are stored in the knowledge base as a set of formulas $F_1$, $F_2$, ..., $F_n$. Facts are represented as atomic formulas, without or with negation, that do not contain variables: $p_i(c_{j1}, c_{j2}, ..., c_{js})$, or $\neg p_i(c_{j1}, c_{j2}, ..., c_{js})$. Facts build up a kind of "database tables". Some of the rules may serve as *integrity conditions*.

c) A **query** is simply another formula ?G. Answering of such a query means that the *query processor* of the knowledge base must determine, does G (or, maybe, $\neg$G) follow from the formulas $F_1$, $F_2$, ..., $F_n$, stored in the knowledge base.

If G contains a free variable x, then the query ?G(x) means the following: return all the object constants $c_i$, for which the formula $G(c_i)$ follows from $F_1$, $F_2$, ..., $F_n$.

**Note**. Two different strategies may be used when building a knowledge base. The so-called **Closed World Assumption** is typical for the traditional databases: the predicate $p_i(c_{j1}, c_{j2}, ..., c_{js})$ is regarded as **true**, if and only if the formula $p_i(c_{j1}, c_{j2}, ..., c_{js})$ is stored in the knowledge base. If there is no such formula in the knowledge base, then $p_i(c_{j1}, c_{j2}, ..., c_{js})$ is regarded as **false**. For example, if the knowledge base does not contain the formula *Father(John, Britney)*, then it is assumed that John is **not** father of Britney.

For knowledge bases more natural is the so-called **Open World Assumption**: if neither the formula $p_i(c_{j1}, c_{j2}, ..., c_{js})$, nor the formula $\neg p_i(c_{j1}, c_{j2}, ..., c_{js})$ is stored in the knowledge base, then the truth-value of the predicate $p_i(c_{j1}, c_{j2}, ..., c_{js})$ is regarded as **unknown**. However, a definite truth value of $p_i(c_{j1}, c_{j2}, ..., c_{js})$ may follow from other formulas stored in the knowledge base. For example, if the knowledge base does not contain neither the formula *Father(John, Britney)*, nor *¬Father(John, Britney),* then it is assumed that

John **is not known** to be (but may be) father of Britney, unless the answer follows from other formulas stored in the database.

Thus, to build the query processor of our knowledge base, we must use some algorithm allowing to determine (as fast as possible), given the formulas $F_1$, $F_2$, ..., $F_n$, G, does G follow from $F_1$, $F_2$, ..., $F_n$, or not. Let's call this task the **reasoning task**.

According to Gödel's Completeness Theorem, G follows from $F_1$, $F_2$, ..., $F_n$, if and only if

$$[\text{L1-L15, MP, Gen}]: F_1, F_2, ..., F_n \vdash G,$$

i.e. if G can be derived from $F_1$, $F_2$, ..., $F_n$ in the classical predicate logic. This makes the reasoning task at least semi-feasible for computers (in the sense of Corollary 4.3.4). However,

**Corollary 4.3.5** (of Church-Kalmar Theorem). If, when building a knowledge base, we will use the full power of some predicate language (containing at least one predicate constant that is at least binary), then the reasoning task will not be algorithmically solvable, and – for such a knowledge base – **we will fail to build a universal query processor**.

Thus, to build a really usable knowledge base, we must restrict somehow our predicate language to make the reasoning task solvable. For a successful attempt to do this see the so-called description logics.

### Skolem's paradox

Initially, the Model Existence Theorem was proved in a weaker form in 1915 (by Leopold Löwenheim) and 1919 (by Thoralf Skolem): if a first order theory has a model, then it has a finite or countable model (the famous **Löwenheim-Skolem theorem**). Proof (after 1949): if T has a model, then T is consistent, i.e. T has a finite or countable model.

**L. Löwenheim.** Ueber Möglichkeiten im Relativkalkül. "Mathematische Annalen", 1915, Vol.76, pp.447-470.

**Th. Skolem.** Logisch-kombinatorische Untersuchungen über die Erfüllbarkeit und Beweisbarkeit mathematischen Sätze nebst einem Theoreme über dichte Mengen. *Videnskabsakademiet i Kristiania, Skrifter* I, No. 4, 1920, pp. 1-36.

Löwenheim-Skolem theorem (and the Model Existence theorem) is steadily provoking the so-called **Skolem's Paradox**, first noted by Skolem in his address before the 5th Congress of Scandinavian Mathematicians (July 4-7, 1922):

**Th. Skolem.** Einige Bemerkungen zur axiomatischen Begründung der Mengenlehre.

*Matematikerkongressen i Helsingfors den 4-7 Juli 1922, Den femte skandinaviska matematikerkongressen, Redogörelse*, Akademiska Bokhandeln, Helsinki, 1923, pp. 217-232.

Skolem called the effect "relativity of set-theoretic notions". In all formal set theories (for example, in ZF) we can prove the existence of uncountable sets. Still, according to the Model Existence theorem, if our formal set theory consistent, then there is a countable model where all its axioms and theorems are true. I.e. a theory proves the existence of uncountable sets, yet it has a countable model! Is this possible? Does it mean that all formal set theories are inconsistent? Platonists put it as follows: any consistent axiomatic set theory has countable models, hence, no axiom system can represent our "intended" set theory (i.e. the Platonist "world of sets") adequately.

For a formalist, Skolem's Paradox is not a paradox at all. I would rather call it Skolem's Effect − like as the photo-effect, it is simply a **striking phenomenon**. Indeed, let J be a countable model of our formal set theory. In this theory, we can prove that the set r of all real numbers is uncountable, i.e.

$$\neg\exists f \, (f \text{ is 1-1 function from r into w}), \qquad (1)$$

where w is the set of all natural numbers. What is the meaning of this theorem in the countable model J? Interpretations of r and w are subsets of the domain $D_J$, i.e. they both are countable sets, i.e.

$$\exists f \, (f \text{ is 1-1 function from } r_J \text{ into } w_J). \qquad (2)$$

Interpretation of (1) in J is

$$\neg\exists f((\quad f \in D_J \quad) \text{ and (f is 1-1 function from } r_J \text{ into } w_J)).$$

Hence, the mapping f of (2) does exist, yet it exists **outside the model** J! Do you think that f of (2) "must" be located in the model? Why? If you are living (as an "internal observer") within the model J, the set $r_J$ seems uncountable to you (because you cannot find a 1-1 function from $r_J$ into $w_J$ in your world J). Still, for me (an "external observer") your uncountable $r_J$ is countable − in my world I have a 1-1 function from $r_J$ into $w_J$!

Hence, indeed, Skolem's Paradox represents simply a *striking phenomenon*. It is worth of knowing, yet there is no danger in it.

Added February 9, 2007.

The inter-relationship of the Completeness Theorem and Model Existence Theorem can be represented in the following very general way.

Let us replace:

- Predicate language L − by any set S of "formulas".

- First order theory T − by any "formula" of S (assume, T contains only a finite number of

axioms, and take the conjuction of them).

- The notion of interpretation − by any set M and a "predicate" T(m, F) (where m is member of M, and F − a formula of S). If you wish, you may read T(m, F) as "m makes F true", i.e. m is a "model" of F.

- The notion of provability in the classical logic − by a "predicate" P(F) (where F is a formula of S). If you wish, you may read P(F) as "F is provable in the classical logic".

Assume, for a a set of "formulas" S, we have any set M and any two "predicates" T(m, F) and P(F) (where m is a member of M, and F − a formula of S) such that only the following simple principles hold:

a) For all F,   $F \in S \rightarrow \neg \neg F \in S$    (i.e. S is closed under negation).

b) For all   $m \in M$   and   $F \in S$  , T(m, F) $\leftrightarrow \neg$T(m, ¬F).

c) For all   $F \in S$  , $\neg\neg$P($\neg\neg$F) → P (F).

**"Completeness Theorem"**. For all F, $\forall$m T(m, F) → P(F).

**"Model Existence Theorem"**. For all F, ¬P(¬F) → $\exists$m T(m, F).

**Theorem.** If a, b, c) hold, then the above "theorems" are equivalent.

**Proof**. 1) Assume $\forall$m T(m, F) → P(F) for all F. Then ¬P(F) → ¬$\forall$m T(m, F), and by a) also, ¬P(¬F) → ¬$\forall$m T(m, ¬F) → $\exists$m¬T(m, ¬F) → $\exists$m T(m, F) by b). Q.E.D.

2) Assume ¬P(¬F) → $\exists$m T(m, F) for all F.Then ¬$\exists$m T(m, F) → ¬¬P(¬F), and by a) also ¬$\exists$m T(m, ¬F) → ¬¬P(¬¬F). By b), $\forall$m T(m, F) → $\forall$m ¬T(m, ¬F) → ¬$\exists$m T(m, ¬F) → ¬¬P(¬¬F) → P(F) by c). Q.E.D.

# 4.4. Constructive Propositional Logic – Kripke Semantics

<u>Saul Aaron Kripke</u>: "American logician and philosopher Saul Kripke is one of today's leading thinkers on thought and its manifold relations to the world. His name is attached to objects in several fields of logic from Kripke-Platek axioms in higher recursion theory to the "Brouwer-Kripke scheme" in intuitionistic mathematics. Kripke models for modal logic, a discovery he made in his teen-age years, became part of the standard vocabulary of mathematical logicians after his first article appeared in 1963, when he was just 23 years old. Kripke models and the results that depend upon them are cited today not only in philosophy and logic, but also in linguistics and computer science..." (<u>The Gazette. The newspaper of the John Hopkins University, May 12, 1997, Vol.26, N 34</u>)

**S. Kripke** (1963). Semantical Considerations on Modal Logic, *Acta Philosophica Fennica* **16**: 83-94.

**S. Kripke** (1963). Semantical analysis of modal logic. I. Normal modal propositional calculi. *Z. Math. Logik Grundl. Math.*, 9:67-96, 1963.

**S. Kripke** (1965). Semantical analysis of intuitionistic logic. In: *J. N. Crossley, M. A. E. Dummet (eds.), Formal systems and recursive functions*. Amsterdam, North Holland, 1965,

pp.92-129.

As usual, let us assume, the formula F has been built of "atomic" formulas $B_1$, $B_2$, ..., $B_n$ by using propositional connectives only. Instead of simply computing truth values of F from truth values of $B_1$, $B_2$, ..., $B_n$, Kripke proposed to consider the **behavior** of F when the truth values of $B_1$, $B_2$, ..., $B_n$ are changing gradually **from false to true** according to some "scenario".

Thus, Kripke proposed to replace the classical semantics (interpretation) of the propositional connectives (defined by the classical truth tables) by a more complicated "dynamic" semantics.

Instead of simply saying that ¬F is true, iff F is false, let us say that, at some point in a scenario, ¬F is true, if and only if, at this point, F is false and remains false, when truth values of $B_1$, $B_2$, ..., $B_n$ are changing according to the scenario.

Let o stand for implication, conjunction or disjunction. Instead of simply saying that FoG is true, if and only if, FoG is true according to the classical truth tables, let us say that, at some point in a scenario, FoG is true, if and only if, at this point, it is true and remains true, when truth values of $B_1$, $B_2$, ..., $B_n$ are changing according to the scenario.

**Example 4.4.1.** Let us consider the behavior of the classical axiom $L_{11}$: $B \lor \neg B$ in the scenario, where, at first, B is false, and at the next step it becomes true:

$$0 \text{ -------------- } 1$$

At the starting point, B is false, ¬B also is false (here, for ¬B to be true, B must remain false at the next step, but it doesn't). This means that, at the starting point, $B \lor \neg B$ is false. At the next step: B is true, hence, ¬B is false, but, of course, $B \lor \neg B$ is true. Thus, in Kripke scenarios, $B \lor \neg B$ is **not** true. Surprisingly, some time later ([Lemma 4.4.3](#)), we will derive from this simple fact that Bv¬B cannot be proved in the constructive logic (we already know this from [Section 2.8](#)).

**Example 4.4.2.** Let us consider the behavior of the (only) classically provable half of the First de Morgan Law: $\neg(A \land B) \to \neg A \lor \neg B$ in the scenario, where, at first A and B are both false, and at the next step, two branches appear in the scenario: in the first branch: A remains false, and B becomes true, in the second branch: A becomes true, and B remains false:

$$\vdash\!-01$$
$$00\text{-}\vdash\!--------$$
$$\vdash\!-10$$

At the starting point: A is false, ¬A – also is false (for ¬A to be true, A must remain false at the next step, but in the second branch it doesn't). Similarly, at the starting point: B is false, ¬B – also false (for ¬B to be true, B must remain false at the next step, but in the first branch it doesn't). This means that, at the starting point, $\neg(A \wedge B)$ is true (because $A \wedge B$ is false, and it remains false in both branches), and $\neg A \vee \neg B$ is false, hence, $\neg(A \wedge B) \rightarrow \neg A \vee \neg B$ is false. Thus, in Kripke scenarios, $\neg(A \wedge B) \rightarrow \neg A \vee \neg B$ is **not** true. Surprisingly, some time later (Lemma 4.4.3), we will derive from this simple fact that the this half of the First de Morgan Law cannot be proved in the constructive logic. We failed to do this in Section 2.8!

**Exercise 4.4.1.** Investigate, in appropriate scenarios, the behavior of the following (only) classically provable formulas:

$$\neg\neg(A \vee B) \rightarrow \neg\neg A \vee \neg\neg B \quad,$$
$$(A \rightarrow B) \rightarrow ((\neg A \rightarrow B) \rightarrow B),$$
$$(A \rightarrow B) \vee (B \rightarrow A) \quad,$$

and verify that, in Kripke scenarios, these formulas are **not** true. Some time later (Lemma 4.4.3), we will derive from this simple fact that these formulas cannot be proved in the constructive logic. We failed to do this in Section 2.8! (Hint: try the most simple scenarios first: 00--01, 00-10, 00-11, etc.)

More precisely, the definition of the **Kripke semantics** for the propositional logic is as follows. Assume, the formula F has been built of "atomic" formulas $B_1$, $B_2$, ..., $B_n$ by using propositional connectives only. Instead of considering truth values of F for all the possible tuples of truth values of $B_1$, $B_2$, ..., $B_n$, let us consider the behavior of F in all the possible Kripke scenarios, defined as follows.

**Definition of Kripke scenarios**. Each scenario s is a triple (b, ≤, t) of the following objects. First, b is a finite set of objects called nodes (or, states).

The second member ≤ is a partial ordering relationship between the nodes, i.e. for all $x, y, z \in b$ : x≤y → (y≤z → x≤z) (transitivity).

The third member t of the tripple is a function (t means "true"). It associates with each node x a "growing" set t(x) of "atomic" formulas, i.e. a subset of $\{B_1, B_2, ..., B_n\}$ in such a way that for all $x, y \in b$ : $x \leq y \rightarrow t(x) \subseteq t(y)$ .

**Note.** In some other textbooks, Kripke scenarios are called **Kripke models**, or **Kripke structures**.

Let us say that $B_i$ is true at the node x, if and only if $B_i$ is in the set t(x). We will denote this fact by x |= $B_i$ ("at x, $B_i$ is true", or "x forces $B_i$"). Since t is

monotonic, if x |= B$_i$ , then y |= B$_i$ for all y after x, i.e. for all $y \in b$ such that y≥x. I.e. if B$_i$ is true at some node x, then B$_i$ remains true at all nodes after x.

Let us define x |= F ("F is true at x", or "x forces F") for any formula F that has been built of "atomic" formulas B$_1$, B$_2$, ..., B$_n$ by using propositional connectives only.

1. Negation. Suppose, the truth value of x |= F is already defined for all $x \in b$ . Then x |= ¬F is defined to be true, if and only if, for all $y \geq x \in b$ , y |= F is false (i.e. ¬(y |= F) is true according to the classical truth table of the negation connective).

2. Implication, conjunction or disjunction. Suppose, the truth values of x |= F and x |= G are already defined for all $x \in b$ . Then x |= FoG is defined to be true, if and only if, for all $y \geq x \in b$ , (y |= F)o(y |= G) is true according to the classical truth table of the implication, conjunction or disjunction connective respectively.

**Lemma 4.4.1.** For any formula F, any Kripke scenario (b, ≤, t), and any node $x \in b$ : if x |= F, then y |= F for all $y \in b$ such that y≥x. I.e. if, in a Kripke scenario, a formula becomes true at some node, then it remains true forever after this node.

**Proof.** By induction.

**Induction base**. See above: if x |= B$_i$ , then y |= B$_i$ for all y after x, i.e. for all $y \in b$ such that y≥x.

**Induction step.**

**1. Negation.** Assume, x |= ¬F, i.e., according to the classical truth table, not y |= F for all $y \geq x \in b$ . If y≥x, then is y |= ¬F true or false? By definition, y |= ¬F would be true, if and only if not z |= F for all $z \geq y \in b$ . By transitivity of ≤, if z≥y and y≥x, then z≥x. Thus, by our assumption, if z≥y, then not z |= F. Hence, y |= ¬F. Q.E.D.

**2. Implication, conjunction or disjunction.** Assume, x |= FoG, i.e., according to the corresponding classical truth table, (y |= F)o(y |= G) is true for all $y \geq x \in b$ . If y≥x, then is y |= FoG true or false? By definition, y |= FoG would be true, if and only if (z |=F)o(z |= G) would be true for all $z \geq y \in b$ . By transitivity of ≤, if z≥y and y≥x, then z≥x. Thus, by our assumption, if z≥x, then (z |= F)o(z |= G) is true. Hence, y |= FoG. Q.E.D.

**Exercise 4.4.2.** Verify that if x is a maximal node in a scenario (b, ≤, t), then x |= F, if and only if F is true at x according to the classical truth tables.

Kripke established that **a formula is provable in the constructive propositional logic, if and only if it is true at all nodes in all Kripke**

**scenarios.**

**Theorem 4.4.2 (S. Kripke, completeness of the constructive propositional logic).** A formula F is provable in the constructive propositional logic (i.e. $[L_1$-$L_{10}$, MP] $\vdash$ F), if and only if F is true at all nodes in all Kripke scenarios.

As usual, the hard part of the proof is establishing that "true is provable", i.e. if F is true at all nodes in all Kripke scenarios, then $[L_1$-$L_{10}$, MP] $\vdash$ F (see Corollary 4.4.7 below). The easy part of the proof is, as usual, the soundness lemma:

**Lemma 4.4.3.** If $[L_1$-$L_{10}$, MP] $\vdash$ F, then F is true at all nodes in all Kripke scenarios.

This lemma will follow from

**Lemma 4.4.4.** If F is any of the constructive axioms $L_1$-$L_{10}$, then, for any Kripke scenario (b, $\leq$, t), and any node $x \in b$ : x |= F. I.e. the constructive axioms are true at all nodes in all Kripke scenarios.

and

**Lemma 4.4.5.** If, in a Kripke scenario (b, $\leq$, t), at the node $x \in b$ : x |= F and x |= F→G, then x |= G. Hence, if F and F→G are true at all nodes in all Kripke scenarios, then so is G.

**Proof of Lemma 4.4.3.** Indeed, by Lemma 4.4.4, all the constructive axioms $L_1$-$L_{10}$ are true at all nodes in all scenarios, and, by Lemma 4.4.5, the Modus Ponens rule preserves the property of being "true at all nodes in all scenarios". Q.E.D.

**Note.** Let us return to the above Example 4.4.2 and Exercise 4.4.1. We established that formulas

$$\neg(A \wedge B) \rightarrow \neg A \vee \neg B \ ;$$
$$\neg\neg(A \vee B) \rightarrow \neg\neg A \vee \neg\neg B \ ;$$
$$(A \rightarrow B) \rightarrow ((\neg A \rightarrow B) \rightarrow B)$$

are not true at all nodes in all scenarios. Hence, by Lemma 4.4.3, these formulas cannot be proved in the constructive logic $[L_1$-$L_{10}$, MP]. We failed to prove this in Section 2.8!

**Proof of Lemma 4.4.5.** We know that x |= F→G means that (y |= F)→(y |= G) is true (according to the classical truth table) for all $y \geq x \in b$ . By Lemma 4.4.1, we know that y |= F for all $y \geq x \in b$ . Hence, if y |= G would be false, then (y |= F)→(y |= G) also would be false. Hence, x |= G. Q.E.D.

**Proof of Lemma 4.4.4.**

**L$_1$: B→(C→B)**

x |= B→(C→B) is true, if and only if (y |= B)→(y |= C→B) is true for all y≥x.

x |= B→(C→B) is false, if and only if (y |= B)→(y |= C→B) is false for some y≥x.

How could (y |= B)→(y |= C→B) be false for some y≥x? According to the classical implication truth table, this could be only, if and only if y |= B is true, and y |= C→B is false.

y |= C→B is true, if and only if (z |= C)→(z |= B) is true for all z≥y.

y |= C→B is false, if and only if (z |= C)→(z |= B) is false for some z≥y.

How could (z |= C)→(z |= B) be false for some z≥y? According to the classical implication truth table, this could be, if and only if z |= C is true, and z |= B is false.

Summary:

$$x |= B→(C→B) \text{ is false}$$
$$\text{if and only if}$$
$$∃y≥x \text{ (}\mathbf{y |= B \text{ is true}} \text{ and } y |= C→B \text{ is false)}$$
$$\text{if and only if}$$
$$∃z≥y \text{ (}z |= C \text{ is true and } \mathbf{z |= B \text{ is false}})$$

Hence, if x |= B→(C→B) is false, then there are y and z such that: x≤y≤z, **y |= B is true**, z |= C is true, and **z |= B is false**. By <u>Lemma 4.4.1</u>, if y≤z and y |= B is true, then z |= B is true. Contradiction with "z |= B is false". Thus, x |= B→(C→B) is true.

**L$_{10}$: ¬B→(B→C)**

x |= ¬B→(B→C) is false, if and only if (y |= ¬B)→(y |= B→C) is false for some y≥x, i.e. if and only if y |= ¬B is true, and y |= B→C is false.

y |= ¬B is true, if and only if z |=B is false for all z≥y.

y |= B→C is false, if and only if (z |= B)→(z |= C) is false for some z≥y, i.e. if and only if z |= B is true, and z |= C is false.

Summary:

$$x |= ¬B→(B→C) \text{ is false}$$
$$\text{if and only if}$$
$$∃y≥x \text{ (}y |= ¬B \text{ is true and } y |= B→C \text{ is false)}$$

| if and only if | if and only if |
|---|---|
| **∀z≥y (z |=B is false)** | **∃z≥y (z |= B is true** and z |= C is false) |

Hence, if x |= ¬B→(B→C) is false, then there is y≥x such that: a) **∀z≥y (z |=B**

**is false**), and b) $\exists z{\geq}y$ (**z |= B is true).** Contradiction. Thus, x |= ¬B→(B→C) is true.

**L₃:** $B \wedge C \rightarrow B$

$$x \models B \wedge C \rightarrow B \text{ is false}$$
$$\text{if and only if}$$
$$\exists y{\geq}x \text{ (y} \models B \wedge C \text{ is true and } \textbf{y} \models \textbf{B is false})$$
$$\text{if and only if}$$
$$\forall z{\geq}y \text{ (} \textbf{z} \models \textbf{B is true} \text{ and z} \models \text{C is true)}$$

Hence, there is y such that x≤y and y |= B is false. From $\forall z{\geq}y$ (z |=B is true) we obtain that y |= B is true. Contradiction. Thus, x |= $B \wedge C \rightarrow C$ is true.

**L₄:** $B \wedge C \rightarrow C$

Similarly.

**L₅:** $B \rightarrow (C \rightarrow B \wedge C)$

$$x \models B \rightarrow (C \rightarrow B \wedge C) \text{ is false}$$
$$\text{if and only if}$$
$$\exists y{\geq}x \text{ (} \textbf{y} \models \textbf{B is true} \text{ and y} \models C \rightarrow B \wedge C \text{ is false)}$$
$$\text{if and only if}$$
$$\exists z{\geq}y \text{ (} \textbf{z} \models \textbf{C is true} \text{ and z} \models B \wedge C \text{ is false)}$$

Hence, there are y, z such that x≤y≤z, y |= B is true, and z |= C is true, and **z |= $B \wedge C$ is false**. Then, by [Lemma 4.4.1](#), (u |= B is true)and(u |= C) for all u≥z. I.e. **z |= $B \wedge C$ is true**. Contradiction. Thus, x |= $B \rightarrow (C \rightarrow B \wedge C)$ is true.

**L₆:** $B \rightarrow B \vee C$

$$x \models B \rightarrow B \vee C \text{ is false}$$
$$\text{if and only if}$$
$$\exists y{\geq}x \text{ (} \textbf{y} \models \textbf{B is true} \text{ and y} \models B \vee C \text{ is false)}$$
$$\text{if and only if}$$
$$\exists z{\geq}y \text{ (} \textbf{z} \models \textbf{B is false} \text{ and z} \models \text{C is false)}$$

Hence, there are y, z such that x≤y≤z, y |= B is true, and z |= B is false. By Lemma 4.4.1, this is a contradiction. Thus, x |= $B \rightarrow B \vee C$ is true.

**L₇:** $C \rightarrow B \vee C$

Similarly.

**L₈:** $(B \rightarrow D) \rightarrow ((C \rightarrow D) \rightarrow (B \vee C \rightarrow D))$

$$x \models (B \rightarrow D) \rightarrow ((C \rightarrow D) \rightarrow (B \vee C \rightarrow D)) \text{ is false}$$
$$\text{if and only if}$$

$\exists y\geq x$ (**y** |=**B→D is true** and y |= $\left(C\rightarrow D\right)\rightarrow\left(B\vee C\rightarrow D\right)$ is false)

if and only if

$\exists z\geq y$ (**z** |= **C→D is true** and z |= $B\vee C\rightarrow D$ is false)

if and only if

$\exists u\geq z$ (u |= $B\vee C$ is true and **u** |= **D is false**)

Hence, there are y, z, u such that x≤y≤z≤u, y |= B→D is true, z |= C→D is true, and u |= D is false. By Lemma 4.4.1, u |= B→D is true, and u |= C→D is true. Thus, if u |= B would be true, then u |= D also would be true. Hence, u |= B is false. Similarly, u |= C also is false. Hence, u |= $B\vee C$ is false. But we know that it is true. Contradiction. Thus, x |= $L_8$ is true.

**$L_2$: (B→(C→D))→((B→C)→(B→D))**

x |= (B→(C→D))→((B→C)→(B→D)) is false

if and only if

$\exists y\geq x$ (y |= B→(C→D) is true and y |= (B→C)→(B→D) is false)

if and only if                                    if and only if

$\forall z\geq y$ ((z |= B)→(z |= C→D))

$\exists z\geq y$ (z |= B→C is true and z |= B→D is false)

if and only if                                    if and only if

$\forall u\geq z$ ((u |= B)→(u |= C))      $\exists u\geq z$ (u |= B is true and u |= D is false)

Hence, there are y, z, u such that x≤y≤z≤u, u |= B is true and **u |= D is false**. From $\forall u\geq z$ ((u |= B)→(u |= C)) we obtain that u |=C also is true, and from $\forall z\geq y$ ((z |= B)→(z |= C→D)) – that z |= C→D is true. Then, by Lemma 4.4.1, u |= C→D also is true, i.e. $\forall v\geq u$ ((v |= C)→(v |= D)), in particular, (u |= C)→(u |= D). Hence, **u|= D is true**. Contradiction. Thus, x |= $L_2$ is true.

**$L_9$: (B→C)→((B→¬C)→¬B)**

x |= (B→C)→((B→¬C)→¬B) is false

if and only if

$\exists y\geq x$ (y |= B→C is true and y |= (B→¬C)→¬B is false)

if and only if                                    if and only if

**$\forall z\geq y$ ((z |= B)→(z |= C))**      $\exists z\geq y$ (z |= B→¬C is true and z |= ¬B is false)

if and only if                                    if and only if

**$\forall u\geq z$ ((u |= B)→(u |= ¬C))**   $\exists u\geq z$ (**u |= B is true**)

Hence, there are y, z, u such that x≤y≤z≤u , and **u |= B is true**. From **$\forall z\geq y$ ((z |= B)→(z |= C))** we obtain that u |= C is true. From **$\forall u\geq z$ ((u |= B)→(u |= ¬C))** we obtain that u |= ¬C is true, i.e. v |= C is false for some v≥u. By Lemma 4.4.1, if u |= C is true, then v |= C is true. Contradiction with "v |= C is false". Hence, x |= $L_9$ is true.

**Exercise 4.4.3.** Verify that, in the above recursive definition of x |= F, the item

2. Implication, conjunction or disjunction: x |= FoG is defined to be true, if and only if, according to the classical truth tables, (y |= F)o(y |= G) is true for all $y \geq x \in b$ .

could be replaced by

2a. Implication ("non-monotonic" connective): x |= F→G is defined to be true, if and only if, according to the classical truth tables, (y |= F)→(y |= G) is true for all $y \geq x \in b$ .

2b. Conjunction or disjunction ("monotonic" connectives): x |= FoG is defined to be true, if and only if, according to the classical truth tables, (x |= F)o(x |= G) is true.

End of Exercise 4.4.3.


**The hard part of the proof**

Now, let us try proving that, if F is true at all nodes in all Kripke scenarios, then F is provable in the constructive propositional logic). We will follow a paper by

**Judith Underwood.** A constructive Completeness Proof for Intuitionistic Propositional Calculus. TR-90-1179, December 1990, *Department of Computer Science, Cornell University.*

based on a constructions from

[Melvin Fitting](). Intuitionistic Logic, Model Theory and Forcing. North-Holland, Amsterdam, 1969

The smart idea is to **generalize the problem** in the following way. Instead of considering constructive provability of single formulas, let us consider the constructive provability of $D_1$, $D_2$, ..., $D_m \vdash C_1 \vee C_2 \vee ... \vee C_n$ for arbitrary formulas $D_1$, $D_2$, ..., $D_m$, $C_1$, $C_2$, ..., $C_n$, i.e. let us consider ordered pairs of sets ($\{D_1, D_2, ..., D_m\}$, $\{C_1, C_2, ..., C_n\}$). Let us call such pairs **sequents**. If $S_1$, $S_2$ are sets of formulas ($S_1$ may be empty), let us call the sequent ($S_1$, $S_2$) constructively provable, if and only if [$L_1$-$L_{10}$, MP]: $S_1 \vdash VS_2$, where $VS_2$ denotes the disjunction of formulas contained in $S_2$. Moreover, let us consider **sets of sequents**. This will allow to carry out a specific induction argument (considering single formulas or single sequents does not allow such an argument!).

Let us say that a Kripke scenario (b, ≤, t) contains a **counterexample** for the sequent ($S_1$, $S_2$), if and only if the sequent is false at some node in the scenario (or, more precisely, if and only if there is $x \in b$ such that x |= F for all

formulas $F \in S_1$ and not x $\models$ G for all formulas $G \in S_2$ ).

Additionally, let us use Corollary 6.1.2(b) of Theorem 6.1.1 to replace all negations ¬F by F→f, where f is an atomic formula, which is "always false", i.e. which, in a sequent $(S_1, S_2)$, never belongs to $S_1$. Thus, formulas mentioned in the proof of the following Theorem 4.4.6 do not contain negations (but they may contain the specific atomic formula f).

**Theorem 4.4.6.** For any set S of sequents, either some sequent of S is constructively provable, or there is a Kripke scenario (b, ≤, t), which contains counterexamples for each sequent in S.

**Proof.** Let us start with a **proof overview**. We will consider the following cases:

**Case 1.** S contains $(S_1, S_2)$ such that $A \wedge B \in S_1 \wedge \neg (A \in S_1 \wedge B \in S_1)$ . Let us consider the set S' obtained from S by adding the "missing" formulas A, B to $S_1$, i.e. by replacing $(S_1, S_2)$ by ( $S_1 \cup \{A, B\}$ , $S_2$). Let us verify that if Theorem is true for S', then it is true for S...

**Case 2.** S contains $(S_1, S_2)$ such that $A \wedge B \in S_2 \wedge \neg (A \in S_2 \vee B \in S_2)$ . Let us consider the following two sets: a) S' – obtained from S by adding the formula A to $S_2$, i.e. by replacing $(S_1, S_2)$ by $(S_1, S_2 \cup \{A\}$ ). b) S" – obtained from S by adding the formula B to $S_2$, i.e. by replacing $(S_1, S_2)$ by $(S_1, S_2 \cup \{B\}$ ). Let us verify that if Theorem is true for S' and S", then it is true for S...

**Case 3.** S contains $(S_1, S_2)$ such that $A \vee B \in S_1 \wedge \neg (A \in S_1 \vee B \in S_1)$ . Let us consider the following two sets: a) S' – obtained from S by adding the formula A to $S_1$, i.e. by replacing $(S_1, S_2)$ by ( $S_1 \cup \{A\}$ , $S_2$). b) S" – obtained from S by adding the formula B to $S_1$, i.e. by replacing $(S_1, S_2)$ by ( $S_1 \cup \{B\}$ , $S_2$). Let us verify that if Theorem is true for S' and S", then it is true for S...

**Case 4.** S contains $(S_1, S_2)$ such that $A \vee B \in S_2 \wedge \neg (A \in S_2 \wedge B \in S_2)$ . Let us consider the set S' obtained from S by adding the "missing" formulas A, B to $S_2$, i.e. by replacing $(S_1, S_2)$ by $(S_1, S_2 \cup \{A, B\}$ ). Let us verify that if Theorem is true for S', then it is true for S...

**Case 5.** S contains $(S_1, S_2)$ such that $A \rightarrow B \in S_1 \wedge \neg (A \in S_2 \vee B \in S_1)$ . Let us consider the following two sets: a) S' – obtained from S by adding the formula A to $S_2$, i.e. by replacing $(S_1, S_2)$ by $(S_1, S_2 \cup \{A\}$ ). b) S" – obtained from S by adding the formula B to $S_1$, i.e. by replacing $(S_1, S_2)$ by ( $S_1 \cup \{B\}$ , $S_2$). Let us verify that if Theorem is true for S' and S", then it is true for S...

**Case 6.** S contains (S$_1$, S$_2$) such that $A \rightarrow B \in S_2$ and for every sequent $(T_1, T_2) \in S$, $\neg(S_1 \subseteq T_1 \wedge A \in T_1 \wedge B \in T_2)$. Let us consider the set S' obtained from S by adding the sequent ($S_1 \cup \{A\}$, B) to it. Let us verify that if Theorem is true for S', then it is true for S...

**Case 7.** None of the above cases hold for S. Then, Theorem is true for S – easy to verify...

The first six cases represent the induction argument: proving of Theorem for a sequent set S is reduced to proving it for some other sets – S' and S". By iterating this reduction, we always arrive happily to the Case 7, where Theorem is easy to verify.

Indeed, let us denote by $universe(S_1, S_2)$ the set of all formulas and sub-formulas (of the formulas) contained in $S_1 \cup S_2$. Let us denote by $universe(S)$ the union of the universes of sequents from S.

**Exercise 4.4.4.** Verify that:

a) When, in the Cases 1-5, the sequent (S$_1$, S$_2$) is replaced by some other sequent (T$_1$, T$_2$), then

$$universe(T_1, T_2) \subseteq universe(S_1, S_2) \quad.$$

b) When, in the Case 6, because of the sequent (S$_1$, S$_2$), the sequent $(S_1 \cup \{A\}, B)$ is added to S, then

$$universe(S_1 \cup \{A\}, B) \subseteq universe(S_1, S_2) \quad.$$

c) For a given $universe(S)$, there exist no more than $N = 2^{|universe(S)|+1}$ different sequents (S$_1$, S$_2$) such that $universe(S1, S2) \leq universe(S)$. And, no more than $2^N$ different sets of sequents.

Thus, any chain of iterated Cases 1-6 cannot be longer than $2^N+1$ – either we will arrive at a set of sequents already built at a previous step, or we will arrive at the Case 7.

Now – the proof as it should be.

**Case 1.** S contains (S$_1$, S$_2$) such that $A \wedge B \in S_1 \wedge \neg(A \in S_1 \wedge B \in S_1)$. Let us consider the set S' obtained from S by adding the "missing" formulas A, B to S$_1$, i.e. by replacing (S$_1$, S$_2$) by $(S_1 \cup \{A, B\}, S_2)$.

Let us verify that if Theorem is true for S', then it is true for S.

Assume, some sequent of S' is constructively provable, then it is $(S_1 \cup \{A, B\}, S_2)$ or some other sequent. If it is some other sequent, then it belongs to S, i.e. some sequent of S is constructively provable. If

$(S_1 \cup \{A, B\}, S_2)$ is constructively provable, then so is (S$_1$, S$_2$). Indeed, if $S_1 \cup \{A, B\} \vdash VS_2$ is constructively provable, how to prove S$_1 \vdash VS_2$? Since S$_1$ contains $A \wedge B$, by axioms L$_3$ and L$_3$ we can derive A and B. After this, we can apply the proof of $S_1 \cup \{A, B\} \vdash VS2$. Hence, S$_1 \vdash VS_2$ is constructively provable.

On the other side, if there is a Kripke scenario (b, ≤, t), which contains a counterexample for each sequent in S', then it contains also a counterexample for each sequent in S. Indeed, a sequent in S is either (S$_1$, S$_2$), or some other sequent. If it is some other sequent, then it belongs to S', i.e. (b, ≤, t) contains a counterexample for it. Does (b, ≤, t) contain a counterexample also for (S$_1$, S$_2$)? We know that it contains a counterexample for $(S_1 \cup \{A, B\}, S_2)$, i.e. for some $x \in b$, x |= F for all formulas $F \in S_1 \cup \{A, B\}$ and not x |= G for all formulas $G \in S_2$. Hence, (b, ≤, t) contains a counterexample also for (S$_1$, S$_2$). Q.E.D.

**Case 2.** S contains (S$_1$, S$_2$) such that $A \wedge B \in S_2 \wedge \neg(A \in S_2 \vee B \in S_2)$. Let us consider the following two sets:

a) S' – obtained from S by adding the formula A to S$_2$, i.e. by replacing (S$_1$, S$_2$) by $(S_1, S_2 \cup \{A\})$.

b) S" – obtained from S by adding the formula B to S$_2$, i.e. by replacing (S$_1$, S$_2$) by $(S_1, S_2 \cup \{B\})$.

Let us verify that if Theorem is true for S' and S", then it is true for S.

Assume, some sequent of S' and some sequent of S" is constructively provable. The sequent of S' is $(S_1, S_2 \cup \{A\})$ or some other sequent. If it is some other sequent, then it belongs to S, i.e. some sequent of S is constructively provable. The sequent of S" is $(S_1, S_2 \cup \{B\})$ or some other sequent. If it is some other sequent, then it belongs to S, i.e. some sequent of S is constructively provable. So, let us consider the situation, when $(S_1, S_2 \cup \{A\})$ and $(S_1, S_2 \cup \{B\})$ both are constructively provable.

If S$_1 \vdash A \vee S_2$ and S$_1 \vdash B \vee S_2$ both are constructively provable, how to prove S$_1 \vdash VS_2$ (we know that S$_2$ contains $A \wedge B$)?

By Theorem 2.3.1, conjunction is distributive to disjunction:

$$[L_1\text{-}L_8, MP]: \vdash (A \wedge B) \vee C \leftrightarrow (A \vee C) \wedge (B \vee C).$$

Hence, [L$_1$-L$_8$, MP]: $(A \vee S_2) \wedge (B \vee S_2) \rightarrow (A \wedge B) \vee S_2$. So, let us merge the

proofs of $S_1 \vdash A \vee S_2$ and $S_1 \vdash B \vee S_2$ , and let us append the proof of Theorem 2.3.1. Thus, we have obtained a proof of $S_1 \vdash (A \wedge B) \vee S_2$ .

From <u>Section 2.3</u> we know that in $[L_1\text{-}L_8, \text{MP}]$ disjunction is associative, commutative and idempotent. And, by Replacement Lemma 1(e):

$[L_1\text{-}L_8, \text{MP}]$ A$\leftrightarrow$B $\vdash A \vee C \leftrightarrow B \vee C$ . Since $S_2$ contains $A \wedge B$ , these facts allow, from a proof of $S_1 \vdash (A \wedge B) \vee S_2$ , to derive a proof of $S_1 \vdash V S_2$.

On the other side, if there is a Kripke scenario (b, $\leq$, t), which contains a counterexample for each sequent in S', then it contains also a counterexample for each sequent in S. Indeed, a sequent in S is either $(S_1, S_2)$, or some other sequent. If it is some other sequent, then it belongs to S', i.e. (b, $\leq$, t) contains a counterexample for it. Does (b, $\leq$, t) contain a counterexample also for $(S_1,$ $S_2)$? We know that it contains a counterexample for $(S_1, S_2 \cup \{A\})$ , i.e. for some $x \in b$ , x |= F for all formulas $F \in S_1$ and not x |= G for all formulas $G \in S_2 \cup \{A\}$ . Hence, (b, $\leq$, t) contains a counterexample also for $(S_1, S_2)$. Q.E.D.

If there is a Kripke scenario (b, $\leq$, t), which contains a counterexample for each sequent in S", then it contains also a counterexample for each sequent in S. The argument is similar to the above.

**Case 3.** S contains $(S_1, S_2)$ such that $A \vee B \in S_1 \wedge \neg (A \in S_1 \vee B \in S_1)$ . Let us consider the following two sets:

a) S' – obtained from S by adding the formula A to $S_1$, i.e. by replacing $(S_1,$ $S_2)$ by $(S_1 \cup \{A\}, S_2)$ .

b) S" – obtained from S by adding the formula B to $S_1$, i.e. by replacing $(S_1,$ $S_2)$ by $(S_1 \cup \{B\}, S_2)$ .

Let us verify that if Theorem is true for S' and S", then it is true for S.

Assume, some sequent of S' and some sequent of S" is constructively provable. The sequent of S' is $(S_1 \cup \{A\}, S_2)$ or some other sequent. If it is some other sequent, then it belongs to S, i.e. some sequent of S is constructively provable. The sequent of S" is $(S_1 \cup \{B\}, S_2)$ or some other sequent. If it is some other sequent, then it belongs to S, i.e. some sequent of S is constructively provable. So, let us consider the situation, when $(S_1 \cup \{A\}, S_2)$ and $(S_1 \cup \{B\}, S_2)$ both are constructively provable.

Let us remind Exercise 2.3.2 $[L_1, L_2, L_8, \text{MP}]$: if $A_1, A_2, ..., A_n, B \vdash D$, and

$A_1, A_2, ..., A_n, C \vdash D$, then $A_1, A_2, ..., A_n$, $B \vee C \vdash D$. Thus, if $S_1 \cup \{A\} \vdash VS_2$ and $S_1 \cup \{B\} \vdash VS_2$ both are constructively provable, then (since $S_1$ contains $A \vee B$) so is $S_1 \cup \{B\} \vdash VS_2$.

On the other side, if there is a Kripke scenario (b, ≤, t), which contains a counterexample for each sequent in S', then it contains also a counterexample for each sequent in S. Indeed, a sequent in S is either $(S_1, S_2)$, or some other sequent. If it is some other sequent, then it belongs to S', i.e. (b, ≤, t) contains a counterexample for it. Does (b, ≤, t) contain a counterexample also for $(S_1,$ $S_2)$? We know that it is contains counterexample for $(S_1 \cup \{A\}, S_2)$ , i.e. for some $x \in b$ , x |= F for all formulas $F \in S_1 \cup \{A\}$ and not x |= G for all formulas $G \in S_2$ . Hence, (b, ≤, t) contains a counterexample also for $(S_1,$ $S_2)$. Q.E.D.

If there is a Kripke scenario (b, ≤, t), which contains a counterexample for each sequent in S'', then it is also contains counterexample for each sequents in S. The argument is similar to the above.

**Case 4.** S contains $(S_1, S_2)$ such that $A \vee B \in S_2 \wedge \neg (A \in S_2 \wedge B \in S_2)$ . Let us consider the set S' obtained from S by adding the "missing" formulas A, B to $S_2$, i.e. by replacing $(S_1, S_2)$ by $(S_1, S_2 \cup \{A, B\})$ .

Let us verify that if Theorem is true for S', then it is true for S.

Assume, some sequent of S' is constructively provable, then it is $(S_1, S_2 \cup \{A, B\})$ or some other sequent. If it is some other sequent, then it belongs to S, i.e. some sequent of S is constructively provable. If $(S_1, S_2 \cup \{A, B\})$ is constructively provable, then so is $(S_1, S_2)$. Indeed, if $S_1 \vdash (A \vee B) \vee S_2$ is constructively provable, how to prove $S_1 \vdash VS_2$ (where $S_2$ contains $A \vee B$ )?

From Section 2.3 we know that in $[L_1\text{-}L_8, MP]$ disjunction is associative, commutative and idempotent. And, by Replacement Lemma 1(e):

$[L_1\text{-}L_8, MP]$ $A \leftrightarrow B \vdash A \vee C \leftrightarrow B \vee C$ . Since that $S_2$ contains AvB, these facts allow, from a proof of $S_1 \vdash (A \vee B) \vee S_2$ , to derive a proof of $S_1 \vdash VS_2$.

On the other side, if there is a Kripke scenario (b, ≤, t), which contains a counterexample for each sequent in S', then it contains also a counterexample for each sequent in S. Indeed, a sequent in S is either $(S_1, S_2)$, or some other sequent. If it is some other sequent, then it belongs to S', i.e. (b, ≤, t) contains a counterexample for it. Does (b, ≤, t) contain a counterexample also for $(S_1,$

$S_2$)? We know that it contains a counterexample for $(S_1, S_2 \cup \{A, B\})$ , i.e. for some $x \in b$ , x $\models$ F for all formulas $F \in S_1$ and not x $\models$ G for all formulas $G \in S_2 \cup \{A, B\}$ . Hence, (b, $\leq$, t) contains a counterexample also for $(S_1, S_2)$. Q.E.D.

If there is a Kripke scenario (b, $\leq$, t), which contains a counterexample for each sequent in S", then it contains also a counterexample for each sequent in S. The argument is similar to the above.

**Case 5.** S contains $(S_1, S_2)$ such that $A \rightarrow B \in S_1 \wedge \neg (A \in S_2 \vee B \in S_1)$ . Let us consider the following two sets:

a) S' – obtained from S by adding the formula A to $S_2$, i.e. by replacing $(S_1, S_2)$ by $(S_1, S_2 \cup \{A\})$ .

b) S" – obtained from S by adding the formula B to $S_1$, i.e. by replacing $(S_1, S_2)$ by $(S_1 \cup \{B\}, S_2)$ .

Let us verify that if Theorem is true for S' and S", then it is true for S.

Assume, some sequent of S' and some sequent of S" is constructively provable. The sequent of S' is $(S_1, S_2 \cup \{A\})$ or some other sequent. If it is some other sequent, then it belongs to S, i.e. some sequent of S is constructively provable. The sequent of S" is $(S_1 \cup \{B\}, S_2)$ or some other sequent. If it is some other sequent, then it belongs to S, i.e. some sequent of S is constructively provable. So, let us consider the situation, when $(S_1, S_2 \cup \{A\})$ and $(S_1 \cup \{B\}, S_2)$ both are constructively provable.

We have two proofs: $S_1 \vdash A \vee S_2$ and $S_1$, B $\vdash VS_2$, and we know that $S_1$ contains A→B. How to derive a proof of $S_1 \vdash VS_2$?

Since $S_1$ contains A→B, we have a proof of $S_1$, A $\vdash$ B. Together with $S_1$, B $\vdash VS_2$ this yields a proof of $S_1$, A $\vdash VS_2$. Of course, $VS_2 \vdash VS_2$. Now, let us remind Exercise 2.3.2 [$L_1$, $L_2$, $L_8$, MP]:

If $A_1$, $A_2$, ..., $A_n$, B $\vdash$ D, and $A_1$, $A_2$, ..., $A_n$, C $\vdash$ D, then $A_1$, $A_2$, ..., $A_n$ , $B \vee C$ $\vdash$ D. Thus, $S_1$, $A \vee S_2$ $\vdash VS_2$. Since we have a proof of $S_1 \vdash A \vee S_2$ , we have also a proof of $S_1 \vdash A \vee S_2$ .

On the other side, if there is a Kripke scenario (b, $\leq$, t), which contains a counterexample for each sequent in S', then it contains also a counterexample for each sequent in S. Indeed, a sequent in S is either $(S_1, S_2)$, or some other sequent. If it is some other sequent, then it belongs to S', i.e. (b, $\leq$, t) contains a

counterexample for it. Does (b, ≤, t) contain a counterexample also for (S$_1$, S$_2$)? We know that it contains a counterexample for $(S_1, S_2 \cup \{A\})$ , i.e. for some $x \in b$ , x |= F for all formulas $F \in S_1$ and not x |= G for all formulas $G \in S_2 \cup \{A\}$ . Hence, (b, ≤, t) contains a counterexample also for (S$_1$, S$_2$). Q.E.D.

If there is a Kripke scenario (b, ≤, t), which contains a counterexample for each sequent in S'', then it contains also a counterexample for each sequent in S. The argument is similar to the above.

**Case 6.** S contains (S$_1$, S$_2$) such that $A \rightarrow B \in S_2$ and for every sequent $(T_1, T_2) \in S$ , $\neg(S_1 \subseteq T_1 \wedge A \in T_1 \wedge B \in T_2)$ . Let us consider the set S' obtained from S by adding the sequent $(S1 U A, B)$ to it.

Let us verify that if Theorem is true for S', then it is true for S.

Assume, some sequent of S' is constructively provable, then it is $(S_1 \cup \{A\}, B)$ or some other sequent. If it is some other sequent, then it belongs to S, i.e. some sequent of S is constructively provable. If $(S_1 \cup \{A\}, B)$ is constructively provable, then so is (S$_1$, S$_2$). Indeed, if S$_1$, A ⊢B is constructively provable, then, by Deduction Theorem 1, S$_1$ ⊢A→B, and S$_1$ ⊢VS$_2$ (since S$_2$ contains A→B).

On the other side, if there is a Kripke scenario (b, ≤, t), which contains a counterexample for each sequent in S', then, since S is a subset of S', this scenario contains also a counterexample for each sequent in S.

**Case 7.** None of the above cases hold for S. Hence, for every sequent $(S_1, S_2) \in S$ the following holds:

1) If $A \wedge B \in S_1$ , then $A \in S_1 \wedge B \in S_1$ ,

2) If $A \wedge B \in S_2$ , then $A \in S_2 \vee B \in S_2$ ,

3) If $A \vee B \in S_1$ , then $A \in S_1 \vee B \in S_1$ ,

4) If $A \vee B \in S_2$ , then $A \in S_2 \wedge B \in S_2$ ,

5) If $A \rightarrow B \in S_1$ , then $A \in S_2 \vee B \in S_1$ ,

6) If $A \rightarrow B \in S_2$ , then there is

$(T1, T2) \in S$ such that $S_1 \subseteq T_1 \wedge A \in T_2 \wedge B \in T_2$ .

For this kind of sequent sets we have a very simple situation:

a) If, in some sequent $(S_1, S_2) \in S$ the sets S$_1$, S$_2$ contain the same formula A, then from L$_6$: $A \rightarrow A \vee B$ we can derive easily that [L$_1$-L$_8$, MP]: S$_1$ ⊢

$VS_2$.

b) If the sets $S_1$, $S_2$ are disjoint for all sequents $(S_1, S_2) \in S$ , then we must (and will) build a scenario, containing a counterexample for each sequent in S.

So, let us suppose that the sets $S_1$, $S_2$ are disjoint for all sequents $(S_1, S_2) \in S$ , and let us define the following Kripke scenario (b, $\leq$, t):

b = S,

x$\leq$y must be defined for every two members x, y of b, i.e. for every two sequents $(S_1, S_2)$ and $(T_1, T_2)$ in S. Let us define $(S_1, S_2) \leq (T_1, T_2)$, if and only if $S_1 \subseteq T_1$ . Of course, $'\subseteq'$ is a partial ordering of b.

t must be a monotonic mapping from members of b to sets of atomic formulas. Let us define $t(S_1, S_2)$ as the set of all atomic formulas in $S_1$. Of course, t is monotonic for $'\subseteq'$ . (And, of course, f – our atomic "false", never belongs to $t(S_1, S_2)$).

Thus, (b, $\leq$, t) is a Kripke scenario. Let us prove that it contains a counterexample for each sequent in S. In fact, we will prove that for each sequent $(S_1, S_2) \in S$ , and each formula F:

$$\text{If } F \in S_1 \text{ , then } (S_1, S_2) \models F.$$

$$\text{If } F \in S_2 \text{ , then not } (S_1, S_2) \models F.$$

This will mean that, $(S_1, S_2)$ represents a counterexample for $(S_1, S_2)$.

Of course, our proof will be by induction along the structure of the formula F.

a) F is an atomic formula.

If $F \in S_1$ , then $F \in t(T_1, T_2)$ for every $(T_1, T_2) \in S$ such that $(S_1, S_2) \leq (T_1, T_2)$. Hence, $(S_1, S_2) \models F$.

If $F \in S_2$ , then, since $S_1$ and $S_2$ are disjoint sets, $F \notin S_1$ , and $F \notin t(S_1, S_2)$ , i.e. not $(S_1, S_2) \models F$.

b) F is $A \wedge B$ .

If $F \in S_1$ , then, by (1), $A \in S_1 \wedge B \in S_1$ . Hence, by induction assumption, $(S_1, S_2) \models A$ and $(S_1, S_2) \models B$, i.e., by <u>Exercise 4.4.3</u>, $(S_1, S_2) \models A \wedge B$ .

If $F \in S_2$ , then, by (2), $A \in S_2 \vee B \in S_2$ . If $A \in S_2$ , then, by induction assumption, not $(S_1, S_2) \models A$, i.e., by Exercise 4.4.3, not $(S_1, S_2) \models A \wedge B$ . If $B \in S_2$ – the argument is similar.

c) F is $A \lor B$ .

If $F \in S_1$ , then, by (3), $A \in S_1 \lor B \in S_1$ . If $A \in S_1$ , then, by induction assumption, $(S_1, S_2) \models A$, i.e., by Exercise 4.4.3, $(S_1, S_2) \models A \lor B$ . If $B \in S_1$ – the argument is similar.

If $F \in S_2$ , then, by (4), $A \in S_2 \land B \in S_2$ . By induction assumption, not $(S_1, S_2) \models A$ and not $(S_1, S_2) \models B$, i.e., by Exercise 4.4.3, not $(S_1, S_2) \models A \lor B$ .

d) F is A→B.

d₁) $F \in S_1$ . We must prove that $(S_1, S_2) \models$ A→B, i.e. that $(T_1, T_2) \models$ A→B for each $(T_1, T_2) \in S$ such that $(S_1, S_2) \leq (T_1, T_2)$. So, let us assume that not $(T_1, T_2) \models$ A→B, i.e. that $(U_1, U_2) \models$ A and not $(U_1, U_2) \models$ B for some $(U_1, U_2) \in S$ such that $(T_1, T_2) \leq (U_1, U_2)$.

Since $A \to B \in S_1$ , then also $A \to B \in U_1$ , and, by (5), $A \in U_2 \lor B \in U_1$ . By induction assumption, this means that not $(U_1, U_2) \models$ A or $(U_1, U_2) \models$ B. Contradiction, hence, $(S_1, S_2) \models$ A→B.

d₂) $F \in S_2$ . We must prove that not $(S_1, S_2) \models$ A→B, i.e. that there is $(T_1, T_2) \in S$ such that $(S_1, S_2) \leq (T_1, T_2)$ and $(T_1, T_2) \models$ A and not $(T_1, T_2) \models$ B.

Since $A \to B \in S_2$ , by (6), there is $(T_1, T_2) \in S$ such that $(S_1, S_2) \leq (T_1, T_2)$ and $A \in T_1$ and $B \in T_2$ . By induction assumption, this means that

$(T_1, T_2) \models$ A and not $(T_1, T_2) \models$ B. Q.E.D.

This completes the proof of Theorem 4.4.6.

**Note.** The above proof contains an **algorithm** allowing to find, for each set S of sequents, either a constructive proof of some sequent of S, or a Kripke scenario containing counterexamples for each sequent of S.

**Corollary 4.4.7.** If a formula F is true at all nodes in all scenarios, then

$[L_1\text{-}L_{10}, \text{MP}] \vdash F$ (i.e. F is provable in the constructive propositional logic).

Indeed, let us consider the set of sequents {(0, {F})} consisting of a single sequent (0, {F}), where 0 is empty set. By Theorem 4.4.6, either the sequent (0, {F}) is constructively provable, or there is a Kripke scenario (b, ≤, t), which contains a counterexample for (0, {F}). Since F is true at all nodes in all Kripke scenarios, it cannot have counterexamples; hence, the sequent (0, {F}) (i.e. the formula F) is constructively provable.

Together with Lemma 4.4.3 this Corollary implies the above Theorem 4.4.2 – **Kripke's theorem on the completeness of the constructive propositional logic**: a formula F is true at all nodes in all Kripke scenarios, if and only if F is provable in the constructive propositional logic.

**Corollary 4.4.8 (decidability of the constructive propositional logic).** There is an **algorithm** allowing to determine for any formula F, is this formula provable in the constructive propositional logic $[L_1\text{-}L_{10}, MP]$, or not.

Gerhard Gentzen established this fact in 1934:

**G. Gentzen**. Untersuchungen über das logische Schliessen II. *Mathematische Zeitschrift*, 1934, Vol. 39, pp. 405-431.

**Corollary 4.4.9**. If $F \vee G$ is true at all nodes in all scenarios, then F is true at all nodes in all scenarios, or G is true at all nodes in all scenarios.

**Proof.** Assume, there is a scenario $(b_1, \leq_1, t_1)$ such that $x_1 \models F$ is false for some $x_1 \in b_1$, and a scenario $(b_2, \leq_2, t_2)$ such that $x_2 \models G$ is false for some $x_2 \in b_2$. We may assume that the (node) sets $b_1$ and $b_2$ do not intersect. Let us merge these scenarios by adding a new common starting node $x_0$, where all $B_i$ are false. Then, $x_0 \models F$ is false (Lemma 4.4.1), and $x_0 \models G$ is false (similarly). Hence, according to the classical disjunction truth table, $x_0 \models F \vee G$ is false. But, $x \models F \vee G$ is true. Hence, $x \models F$ is true, or $x \models G$ is true. Q.E.D.

**Theorem 4.4.10.** (Gödel [1932]). If $[L_1\text{-}L_{10}, MP]: \vdash B \vee C$, then

$[L_1\text{-}L_{10}, MP]: \vdash B$ or $[L_1\text{-}L_{10}, MP]: \vdash C$. (I.e. if the disjunction $B \vee C$ is constructively provable, then one of the formulas B, C also is constructively provable.)

**Proof.** If $[L_1\text{-}L_{10}, MP]: \vdash B \vee C$, then, by Kripke's Completeness Theorem 4.4.2, $B \vee C$ is true at all nodes in all scenarios. Then, by Corollary 4.4.9, so is B or so is C. By Kripke's Completeness Theorem 4.4.2, this means that one of the formulas B, C is constructively provable. Q.E.D.

Let us remind the constructive interpretation of disjunction from Section 1.3:

- To prove $B \vee C$ constructively, you must prove B, or prove C. To prove $B \vee C$ classically, you may assume $\neg(B \vee C)$ as a hypothesis, and derive a contradiction. Having only such a "negative" proof, you may be unable to determine, which part of the disjunction $B \vee C$ is true – B, or C, or both.

According to Theorem 4.4.10, the constructive propositional logic $[L_1\text{-}L_{10}, MP]$ supports the constructive interpretation of disjunction.

K.Gödel established this fact in 1932:

**K. Gödel**. Zum intuitionistischen Aussagenkalkül. *Akademie der Wissenschaften in Wien, Mathematisch- naturwissenschaftliche Klasse, Anzeiger*, 1932, Vol.69, pp.65-66.

**Exercise 4.4.5** (optional, for smart students). By adding the schema $(B \rightarrow C) \vee (C \rightarrow B)$ to the axioms of the constructive logic, we obtain the so-called *Gödel-Dummett logic* (Michael Dummett). Verify, that a propositional formula F is provable in Gödel-Dummett logic, if and only if F is true at all nodes in all linear Kripke scenarios (i.e. in the scenarious that do not allow branching). See also Intuitionistic Logic by **Joan Moschovakis** in Stanford Encyclopedia of Philosophy.

# 5. Normal Forms. Resolution Method

In this section, we will try to produce a practical method allowing to derive consequences and prove theorems by using computers. In general, this task is not feasible because of its enormous computational complexity (see Section 4.3). Still, for problems of a "practical size" (arising, for example, in deductive databases and other artificial intelligence systems, or, trying to formalize real mathematical proofs), such methods are possible and some of them are already implemented successfully.

This field of research is called automated reasoning, or automated theorem-proving.

**Warning!** The principal results of this Section are valid only for the **classical logic**!


**Main Ideas**

If $F_1$, ..., $F_n$ is the set of our assumptions (facts, rules, axioms, hypotheses etc.), does the assertion G follow from this set? One of the well known approaches to proving theorems in mathematics – and especially convenient for computers – are the so-called **refutation proofs** (*reductio ad absurdum*) – proofs by deriving a contradiction: assume ¬G, and try deriving a contradiction. I.e. try proving that $F_1$, ..., $F_n$, ¬G is an inconsistent (or, equivalently, unsatisfiable) set of assumptions.

**Idea #1:** let us derive consequences and prove theorems **only in this way**. Let us try developing the best possible method of deriving contradictions from inconsistent sets of assumptions. This (at first glance – trivial) decision is one of the most important steps in the whole story – it will allow (see Section 5.2 below) conversion of the formulas $F_1$, ..., $F_n$, ¬G into a form that does not contain existential quantifiers. And after this, having universal quantifiers only, we will simply drop them all, and continue working with quantifier-free formulas (see Section 5.4).

**Idea #2**: to prove that $F_1$, ..., $F_n$, ¬G is an inconsistent set of formulas, let us "normalize" these formulas as far as possible.

The first step (**idea #2a**) is reducing to the so-called **prenex normal form** – moving all the quantifiers to left. For example, the formula

$$[(\exists x B(x) \to \exists x C(x)) \to \exists x D(x)] \to \exists x F(x)$$

is equivalent (in the classical logic!) to the following formula in prenex normal

form:

$$\forall x_1 \exists x_2 \forall x_3 \exists x_4 [[(B(x_1) \to C(x_2)) \to D(x_3)] \to F(x_4)].$$

(When moving quantifiers to left, some of them must be changed from $\exists$ to $\forall$, or from $\forall$ to $\exists$, see Section 5.1 below.)

The second step (**idea #2b**, due to Thoralf Skolem) allows elimination of existential quantifiers. Indeed, $\forall x_1 \exists x_2$ means that $x_2 = f(x_1)$, and $\forall x_1 \forall x_3 \exists x_4$ means that $x_4 = g(x_1, x_3)$, where f and g are some functions (see Section 5.2). In this way we obtain the so-called **Skolem normal form**, containing universal quantifiers only:

$$\forall x_1 \forall x_3 [[(B(x_1) \to C(f(x_1))) \to D(x_3)] \to F(g(x_1, x_3))].$$

Note that a formula and its Skolem normal form **are not equivalent** (even in the classical logic!), they are only a kind of "semi-equivalent": if a set of formulas allows deriving of contradictions, then the set of their Skolem normal forms allows deriving of contradictions (and conversely).

Now (**idea #2c**), since, our formula contains universal quantifiers only, we may **drop these quantifiers**:

$$[(B(x_1) \to C(f(x_1))) \to D(x_3)] \to F(g(x_1, x_3)).$$

Nothing is lost here. The possibility of deriving contradictions is retained – the quantifiers can be restored by applying the *Gen*-rule.

The third step (**idea #2d**) – reduction of quantifier-free formulas to the so-called **conjunctive normal form** (a conjunction of disjunctions of atomic formulas – with or without negations, see Section 5.3). For example, the above formula can be reduced to the following form:

$$(\neg B(x_1) \vee C(f(x_1)) \vee F(g(x_{1,} x_3))) \wedge (\neg D(x_3) \vee F(g(x_{1,} x_3))) \quad .$$

By assuming that a set of formulas means their conjunction, we can drop the conjunction(s) obtaining a set of the so-called **clauses**:

$$\neg B(x_1) \vee C(f(x_1)) \vee F(g(x_{1,} x_3)) \quad ;$$
$$\neg D(x_3) \vee F(g(x_{1,} x_3)) \quad .$$

Each clause is a disjunctions of atomic formulas – with or without negations. To separate clearly the meaning of each clause, we must rename some of the variables – no two clauses are allowed to contain common variables:

$$\neg B(x_1) \vee C(f(x_1)) \vee F(g(x_{1,} x_3)) \quad ;$$
$$\neg D(x_5) \vee F(g(x_{4,} x_5)) \quad .$$

In this way, instead of our initial set of assumptions $F_1$, ..., $F_n$, ¬G, we obtain a set of separate clauses ("a large cloud of simple disjunctions"), which allows deriving of contradictions, if and only if the initial set $F_1$, ..., $F_n$, ¬G allows it.

The last step – working with the obtained set of clauses.

**Idea #3** (due to <u>John Alan Robinson</u>, see <u>Section 5.5</u> and <u>5.7</u>) – a set of clauses ("large cloud of simple disjunctions") allows deriving of contradictions, if and only if a contradiction can be derived from it by using term substitution and the so-called **Robinson's Resolution rule**:

$$\frac{F \vee \boldsymbol{C}, \neg \boldsymbol{C} \vee G}{F \vee G} \quad .$$

Continue reading...

**An alternative method**: the so-called <u>Method of Analytic Tableaux</u>.

## 5.1. Prenex Normal Form

**Warning!** The principal results of this Section are valid only for the **classical logic**!

Let us consider an interpretation J of some predicate language L, such that the domain $D_J$ contains an infinite set of "objects". Under such interpretation, the "meaning" of formulas containing quantifiers may be more or less non-constructive, or, at least, "constructively difficult".

For example, the formula ∀xB(x) will be true, if B(x) will be true for all "objects" x in the (infinite!) set $D_J$. Thus, it is impossible to verify directly (i.e. "empirically"), is ∀xB(x) true or not. Saying that ∀x∀y(x+y=y+x) is true under the standard interpretation of first order arithmetic, does not mean that we have verified this fact empirically – by checking x+y=y+x for all pairs of natural numbers x, y. Then, how do we know that ∀x∀y(x+y=y+x) is true? Of course, we either postulated this feature of natural numbers directly (i.e. derived it from "empirical evidence"), or proved it by using some set of axioms (i.e. derived it from other postulates). But, in general, formulas having the form ∀xB(x), are "constructively difficult".

The formula ∀x∃yC(x, y) may be even more difficult: it will be true, if for each x in $D_J$ we will be able to find y in $D_J$ such that C(x, y) is true. Thus, thinking constructively, we could say that ∀x∃yC(x, y) is true, only, if there is an algorithm, which, for each x in $D_J$ can find y in $D_J$ such that C(x, y) is true.

For example, under the standard interpretation of first order arithmetic, the formula

$$\forall x \, \exists y \, (x < y \land prime(y))$$

is true (i.e. "there are infinitely many prime numbers"). How do we know this? This fact was proved in VI century BC. But the (similarly quantified) formula

$$\forall x \, \exists y \, (x < y \land prime(y) \land prime(y + 2)) \quad,$$

i.e. the famous twin prime conjecture, is it true or not? Until now, nobody knows the answer.

**Exercise 5.1.1** (optional). Verify that the "meaning" of $\forall x \exists y \forall z D(x, y, z)$ and $\forall x \exists y \forall z \exists u F(x, y, z, u)$ may be even more non-constructive.

But how about the formula $\exists x G(x) \rightarrow \exists y H(y)$? Is it constructively more difficult than $\forall x \exists y C(x, y)$, or less? In general, we could prove that $\exists x G(x) \rightarrow \exists y H(y)$ is true, if we had an algorithm, which, for each $x \in D_J$ such that $G(x)$ is true, could find $y \in D_I$ such that $G(y)$ is true, i.e. if $\forall x \exists y (G(x) \rightarrow H(y))$ would be true. We will establish below, that, in the classical logic, if G does not contain y, and H does not contain x, then the formula $\exists x G(x) \rightarrow \exists y H(y)$ is equivalent to $\forall x \exists y (G(x) \rightarrow H(y))$. Thus, in general, the formula $\exists x G(x) \rightarrow \exists y H(y)$ is constructively as difficult as is the formula $\forall x \exists y C(x, y)$!

To generalize this approach to comparing "constructive difficulty" of formulas, the so-called **prenex normal forms** have been introduced:

a) If a formula does not contain quantifiers, then it is in the prenex normal form.

b) If x is any variable, and the formula F is in the prenex normal form, then $\forall x F$ and $\exists x F$ also are in the prenex normal form.

c) (If you wish so,) there are no other formulas in the prenex normal form.

I.e. a formula is in the prenex normal form, if and only if it has all its quantifiers gathered in front of a formula that does not contain quantifiers. It appears, that in the classical logic, each formula cane be "reduced" to an appropriate equivalent formula in the prenex normal form. To obtain this normal form, the following Lemmas 5.1.1-5.1.3 can be used.

**Lemma 5.1.1.** If the formula G does not contain x as a free variable, then:

a) $[L_1, L_2, L_5, L_{12}, L_{14}, MP, Gen]$: $(G \rightarrow \forall x F(x)) \leftrightarrow \forall x (G \rightarrow F(x))$.

b) $[L_1, L_2, L_5, L_{12}\text{-}L_{15}, MP, Gen]$: $(\exists x F(x) \rightarrow G) \leftrightarrow \forall x (F(x) \rightarrow G)$. What does it mean precisely?

c) $[L_1\text{-}L_{11}, L_{12}\text{-}L_{15}, MP, Gen]$: $(G \rightarrow \exists x F(x)) \leftrightarrow \exists x (G \rightarrow F(x))$. More precisely:

$[L_1\text{-}L_{11}, L_{12}\text{-}L_{15}, MP, Gen]$: $(G\to\exists xF(x)) \to \exists x(G\to F(x))$. This formula cannot be proved constructively! Explain, why. But the converse formula can be proved constructively:

$[L_1, L_2, L_{13}\text{-}L_{15}, MP, Gen]$: $\exists x(G\to F(x)) \to (G\to\exists xF(x))$.

d) $[L_1\text{-}L_{11}, L_{12}\text{-}L_{15}, MP, Gen]$: $(\forall xF(x)\to G) \leftrightarrow \exists x(F(x)\to G)$. What does it mean precisely? More precisely:

$[L_1\text{-}L_{11}, L_{12}\text{-}L_{15}, MP, Gen]$: $(\forall xF(x)\to G) \to \exists x(F(x)\to G)$. This formula cannot be proved constructively! Explain, why. But the converse formula can be proved constructively:

$[L_1, L_2, L_{13}\text{-}L_{15}, MP, Gen]$: $\exists x(F(x)\to G) \to (\forall xF(x)\to G)$.

**Proof.** First, let us note that (a)$\leftarrow$ is an instance of the axiom $L_{14}$, and that (b)$\leftarrow$ is an instance of the axiom $L_{15}$.

Prove (a)$\to$, (b)$\to$, (c)$\leftarrow$, (d)$\leftarrow$ as the Exercise 5.1.2 below.

It remains to prove (c)$\to$ and (d)$\to$ in the classical logic (constructive proofs are impossible).

Let us prove (c)$\to$: $(G\to\exists xF(x)) \to \exists x(G\to F(x))$.

| | First, let us prove: | $\neg G \to ((G\to\exists xF(x))\to\exists x(G\to F(x)))$ |
|---|---|---|
| (1) | $\neg G\to(G\to F(x))$ | Axiom $L_{10}$. |
| (2) | $(G\to F(x))\to\exists x(G\to F(x))$ | Axiom $L_{13}$: $F(x)\to\exists xF(x)$. |
| (3) | $\neg G\to\exists x(G\to F(x))$ | From (1) and (2). |
| (4) | $\neg G \to ((G\to\exists xF(x)) \to \exists x(G\to F(x)))$ | By Axiom $L_1$: $B\to(C\to B)$. |
| | Now, let us prove: | $G \to ((G\to\exists xF(x))\to\exists x(G\to F(x)))$ |
| (5) | $G$ | Hypothesis. |
| (6) | $G\to\exists xF(x)$ | Hypothesis. |
| (7) | $\exists xF(x)$ | From (5) and (6). |
| (8) | $F(x)\to(G\to F(x))$ | Axiom $L_1$: $B\to(C\to B)$. |
| (9) | $\forall x(F(x)\to(G\to F(x)))$ | By Gen. |

| | | |
|---|---|---|
| (10) | $\exists xF(x)\to\exists x(G\to F(x))$ | By Theorem 3.1.1(b), [$L_1$, $L_2$, $L_{12}$-$L_{15}$, MP, Gen] $\vdash$ $\forall x(B\to C)\to(\exists xB\to\exists xC)$. |
| (11) | $\exists x(G\to F(x))$ | From (7) and (10). |
| (12) | $G\to((G\to\exists xF(x))\to$ $\exists x(G\to F(x)))$ | By Deduction Theorem 2 (x is not a free variable in G and $G\to\exists xF(x)$. |
| (13) | **$Gv\neg G\to((G\to\exists xF(x))\to$** $\exists x(G\to F(x)))$ | From (4) and (12), by Axiom $L_8$. The total is [$L_1$, $L_2$, $L_8$, $L_{10}$, $L_{12}$-$L_{15}$, MP, Gen] |
| (14) | $(G\to\exists xF(x)\to\exists x(G\to F(x))$ | By **Axiom $L_{11}$**: $Gv\neg G$. |

Finally, let us prove (d)$\to$: $(\forall xF(x)\to G)\to\exists x(F(x)\to G)$. Let us denote this formula by H.

| | First, let us prove: | **$\forall xF(x)\to H$** |
|---|---|---|
| (1) | $\forall xF(x)$ | Hypothesis. |
| (2) | $\forall xF(x)\to G$ | Hypothesis. |
| (3) | $G$ | From (1) and (2). |
| (4) | $F(x)\to G$ | By Axiom $L_1$: $B\to(C\to B)$. |
| (5) | $\exists x(F(x)\to G)$ | By Axiom $L_{13}$: $F(x)\to\exists xF(x)$. |
| (6) | $\forall xF(x)\to H$ | By Deduction Theorem 2. |
| | Now, let us prove: | **$\exists x\neg F(x)\to H$** |
| (5) | $\neg F(x)$ | Hypothesis. |
| (6) | $\neg F(x)\to(F(x)\to G)$ | Axiom $L_{10}$. |
| (7) | $F(x)\to G$ | From (5) and (6). |
| (8) | $\exists x(F(x)\to G)$ | By Axiom $L_{13}$: $F(x)\to\exists xF(x)$. |
| (9) | $(\forall xF(x)\to G)\to\exists x(F(x)\to G)$ | By Axiom $L_1$: $B\to(C\to B)$. |

| (10)  ¬F(x)→H | By Deduction Theorem 2. |
| (11)  ∃x¬F(x)→H | By Gen and Axiom L$_{15}$: ∀x(¬F(x)→H)→ (∃x¬F(x)→H). |
| (12)  ¬∀xF(x)→H | By <u>Section 3.2</u>, III-4. [L$_1$-L$_{11}$, L$_{13}$, L$_{14}$, MP, Gen]: ⊢ ¬∀xF(x)→∃x¬F(x). **Axiom L$_{11}$ is used here!** |
| (13)  ∀xF(x) v ¬∀xF(x) → H | From (4) and (12), by Axiom L$_8$. |
| (13)  H | By Axiom L$_{11}$: **∀xF(x) v ¬∀xF(x)** |

Q.E.D.

**Exercise 5.1.2.** a) Prove (a)→ of Lemma 5.1.1.

b) Prove (b)→ of Lemma 5.1.1.

c) Prove (c)← of Lemma 5.1.1. (Hint: prove (G→F(x))→(G→∃xF(x)), apply *Gen* and Axiom L$_{15}$.)

d) Prove (d)← of Lemma 5.1.1. (Hint: prove(F(x)→G)→(∀xF(x)→G), apply *Gen* and Axiom L$_{15}$.)

**Lemma 5.1.2.** If the formula G does not contain x as a free variable, then

a) [L$_1$-L$_5$, L$_{13}$-L$_{15}$, MP, Gen]:  $\exists xF(x)\wedge G\leftrightarrow\exists x(F(x)\wedge G)$  .

b) [L$_1$-L$_5$, L$_{12}$, L$_{14}$, MP, Gen]:  $\forall xF(x)\wedge G\leftrightarrow\forall x(F(x)\wedge G)$  .

c) [L$_1$-L$_8$, L$_{13}$-L$_{15}$, MP, Gen]:  $\exists xF(x)\vee G\leftrightarrow\exists x(F(x)\vee G)$  .

d) [L$_1$-L$_8$, L$_{12}$, L$_{14}$, MP, Gen]:  $\forall xF(x)\vee G\leftrightarrow\forall x(F(x)\vee G)$  .

**Proof.**

Prove (a, b, c, d) as the Exercise 5.1.3 below.

Q.E.D

**Exercise 5.1.3.** a) Prove (a, b) of Lemma 5.1.2. (Hint: to prove (a)→, first prove  $F(x)\rightarrow(G\rightarrow\exists x(F(x)\wedge G))$  , and apply Axiom L$_{15}$.)

b) Prove (c) of Lemma 5.1.2. (Hint: to prove (c)→, first prove

$F(x) \rightarrow \exists\, x(F(x) \vee G)$ , apply Gen and Axiom Axiom $L_{15}$; to prove (c)$\leftarrow$ , first prove $F(x) \vee G \rightarrow \exists\, xF(x) \vee G$ , apply Theorem 2.3.1, if necessary.)

c) Prove (d) of Lemma 5.1.2. (Hint: to prove (d)$\rightarrow$, apply Theorem 2.3.1; to prove (d)$\leftarrow$, apply $L_{12}$ and Theorem 2.3.1 to obtain $\forall\, x(F(x) \vee G) \vdash \forall\, xF(x) \vee G$ .)

**Lemma 5.1.3.** a) $[L_1\text{-}L_{10}, L_{12}\text{-}L_{15}, \text{MP, Gen}]$: $\neg\exists xF(x) \leftrightarrow \forall x\neg F(x)$.

b) $[L_1\text{-}L_{11}, L_{12}\text{-}L_{15}, \text{MP, Gen}]$: $\neg\forall xF(x) \leftrightarrow \exists x\neg F(x)$. More precisely:

$[L_1\text{-}L_{11}, L_{13}, L_{14}, \text{MP, Gen}]$: $\neg\forall xF(x) \rightarrow \exists x\neg F(x)$. This formula cannot be proved constructively! Why? But,

$[L_1\text{-}L_{10}, L_{13}, L_{14}, \text{MP, Gen}]$: $\exists x\neg F(x) \rightarrow \neg\forall xF(x)$.

**Proof.**

a) See Section 3.2, Group IV.

b)$\rightarrow$. This is exactly Section 3.2, III-4.

b)$\leftarrow$. See Section 3.2, Group III.

Q.E.D.

Let us remind that a formula is in the prenex normal form, if and only if it has all its quantifiers gathered in front of a formula that does not contain quantifiers.

**Theorem 5.1.4.** In the classical logic, each formula is equivalent to an appropriate formula in the prenex normal form. More precisely, if F is a formula, then, following a simple algorithm, a formula F' can be constructed such that: a) F' is in a prenex normal form, b) F' has the same free variables as F, c) $[L_1\text{-}L_{11}, L_{12}\text{-}L_{15}, \text{MP, Gen}]$: F$\leftrightarrow$F'.

**Proof.** Let us start by an example:

$$\exists xG(x) \rightarrow \exists yH(y).$$

If H did not contain x as a free variable, then, by Lemma 5.1.1(b): $\exists xF(x) \rightarrow G \leftrightarrow \forall x(F(x) \rightarrow G)$, i.e. this formula would be equivalent to $\forall x(G(x) \rightarrow \exists yH(y))$. Now, let us consider the sub-formula $G(x) \rightarrow \exists yH(y)$. If G did not contain y as a free variable, then, by Lemma 5.1.1(c): $G \rightarrow \exists xF(x) \leftrightarrow \exists x(G \rightarrow F(x))$, the sub-formula would be equivalent to $\exists y(G(x) \rightarrow H(y))$. Hence, by Replacement Theorem 2, $\forall x(G(x) \rightarrow \exists yH(y))$ would be equivalent to $\forall x\exists y(G(x) \rightarrow H(y))$.

But, if H would contain x as a free variable, and/or G would contain y as a free variable? Then our "shifting quantifiers up" would be wrong – the formula $\forall x\exists y(G(x) \rightarrow H(y))$ would **not** be equivalent to $\exists xG(x) \rightarrow \exists yH(y)$.

To avoid this problem, let us use Replacement Theorem 3, which says that the meaning of a formula does not depend on the names of bound variables used in it. Thus, as the first step, in $\exists xG(x)$, let us replace x by another variable $x_1$ that does not appear neither in G, nor in H. Then, by Replacement Theorem 3, $\exists xG(x)$ is equivalent to $\exists x_1 G(x_1)$, and by Replacement Theorem 2, $\exists xG(x) \rightarrow \exists yH(y)$ is equivalent to $\exists x_1 G(x_1) \rightarrow \exists yH(y)$. Now, $\forall x_1(G(x_1) \rightarrow \exists yH(y))$ is really equivalent to $\exists x_1 G(x_1) \rightarrow \exists yH(y)$. As the next step, in $\exists yH(y)$, let us replace y by another variable $y_1$ that does not appear neither in G, nor in H. Then, by Replacement Theorem 3, $\exists yH(y)$ is equivalent to $\exists y_1 H(y_1)$, and by Replacement Theorem 2, $G(x_1) \rightarrow \exists y_1 H(y_1)$ is equivalent to $\exists y_1(G(x_1) \rightarrow H(y_1))$. And, finally, $\exists xG(x) \rightarrow \exists yH(y)$ is equivalent to $\forall x_1 \exists y_1(G(x_1) \rightarrow H(y_1))$.

Now, we can start the general proof. In a formula F, let us find the **leftmost** quantifier having a propositional connective over it. If such a quantifier does not exist, the formula is in the prenex normal form. If such a quantifier exists, then F is in one of the following forms:

$$Q_q Q_q ... Q_q(...(\neg QxG)...), \text{ or } Q_q Q_q ... Q_q(...(QxGooH)...), \text{ or } Q_q Q_q ... Q_q(... (GooQxH)...),$$

where $Q_q Q_q ... Q_q$ are the quantifiers "already in prefix", Q is the quantifier in question, and oo is the propositional connective standing directly over Q.

In the first case, by Lemma 5.1.3, $\neg QxG$ is equivalent to $Q'x\neg G$, where Q' is the quantifier opposite to Q. By Replacement Theorem 2, $Q_q Q_q ... Q_q(...(\neg QxG)...)$ is then equivalent to $Q_q Q_q ... Q_q(...(Q'x\neg G)...)$, i.e. Q' has now one propositional connective less over it ( (than had Q).

In the second case, as the first step, in QxG, let us replace x by another variable $x_1$ that does not appear in the entire formula F at all. Then, by Replacement Theorem 3, QxG is equivalent to $Qx_1 G_1$, and by Replacement Theorem 2, $Q_q Q_q ... Q_q(...(QxGooH)...)$ is equivalent to $Q_q Q_q ... Q_q(... (Qx_1 G_1 ooH)...)$. Now, we can apply the appropriate case of Lemma 5.1.1 or Lemma 5.1.2, obtaining that $Qx_1 G_1 ooH$ is equivalent to $Q'x_1(G_1 ooH)$, where Q' is the quantifier determined by the lemma applied. Then , by Replacement Theorem 2, $Q_q Q_q ... Q_q(...(Qx_1 G_1 ooH)...)$ is equivalent to $Q_q Q_q ... Q_q(... (Q'x_1(G_1 ooH))...)$, i.e. Q' has now one propositional connective less over it (than had Q).

In the third case, the argument is similar.

By iterating this operation a finite number of times, we arrive at a formula F' which is in the prenex normal form, and which is (in the classical logic) equivalent to F. Q.E.D.

**Note.** Most formulas admit many different prenex normal forms. For example, the above formula ∃xG(x)→∃xH(x) is equivalent not only to $\forall x_1 \exists x_2 (G(x_1) \to H(x_2))$, but also to $\exists x_2 \forall x_1 (G(x_1) \to H(x_2))$ (verify).

As an example, let us obtain a prenex normal form of the following formula:

$$\exists xB(x) \vee \forall xC(x) \to \forall xD(x) \wedge (\neg \forall xF(x)) \quad.$$

First, assign unique names to bound variables:

$$\exists x_1 B(x_1) \vee \forall x_2 C(x_2) \to \forall x_3 D(x_3) \wedge (\neg \forall x_4 F(x_4)) \quad.$$

Process disjunction:

$$\exists x_1 \forall x_2 (B(x_1) \vee C(x_2)) \to \forall x_3 D(x_3) \wedge (\neg \forall x_4 F(x_4)) \quad.$$

Process negation (∀-∃):

$$\exists x_1 \forall x_2 (B(x_1) \vee C(x_2)) \to \forall x_3 D(x_3) \wedge \exists x_4 \neg F(x_4) \quad.$$

Process conjunction:

$$\exists x_1 \forall x_2 (B(x_1) \vee C(x_2)) \to \forall x_3 \exists x_4 (D(x_3) \wedge \neg F(x_4)) \quad.$$

Process implication premise (∃-∀, ∀-∃):

$$\forall x_1 \exists x_2 (B(x_1) \vee C(x_2) \to \forall x_3 \exists x_4 (D(x_3) \wedge \neg F(x_4))) \quad.$$

Process implication consequent:

$$\forall x_1 \exists x_2 \forall x_3 \exists x_4 (B(x_1) \vee C(x_2) \to D(x_3) \wedge \neg F(x_4)) \quad.$$

The last two steps could be performed in the reverse order as well.

**Warning!** The step "assign unique names to bound variables" is essential! Without it, senseless results would be obtained. For example, converting of ∃xG(x)→∃xH(x) into ∀x∃x(G(x)→H(x)) **is senseless**.

**Exercise 5.1.4.** Transform each of the following formulas into a prenex normal form. Write down every single step of the process. (Hint: the algorithm is explained in the proof of Theorem 5.1.4.)

a) $\exists xB(x) \to (\exists xC(x) \to \exists xD(x))$ ,

b) $\forall x \exists yB(x,y) \wedge \exists xC(x) \to \forall y \exists xD(x,y)$ ,

c) $\exists xB(x,y,z) \to \forall xC(x,y) \vee \exists yD(y,z)$ ,

d) $\forall xB(x) \to (\forall xC(x) \to (\forall xD(x) \to \forall xF(x)))$ ,

e) $((\exists xB(x) \to \exists xC(x)) \to \exists xD(x)) \to \exists xF(x)$ .

**Note.** From a programmer's point of view, prenex normal forms are, in a sense, a crazy invention. In computer programming, you always try to reduce loop bodies, not to extend them as much as possible!

**Exercise 5.1.5** (optional, for smart students). We may use reduction to prenex normal forms in proofs. More precisely, let us try extending the classical logic by introducing of the following additional inference rule (let us call it **PNF-rule**): given a formula F, replace it by some its prenex normal form F'. Verify, that, in fact, this rule does not extend the classical logic, i.e. if there is a proof of $F_1$, $F_2$, ..., $F_n \vdash G$ in $[L_1\text{-}L_{15}$, MP, Gen, PNF-rule], then there is a proof of the same in $[L_1\text{-}L_{15}$, MP, Gen]. (In some other texts, such rules are called **admissible rules**. Thus, the PNF-rule is an admissible rule in the classical logic.)

The notion of prenex normal forms and a version of Theorem 5.1.4 were known to Charles S. Peirce in 1885:

**C. S. Peirce.** On the algebra of logic: A contribution to the philosophy of notation. *American Journal of Mathematics*, 1885, vol.7, pp.180-202.

As noted by **Alasdair Urquhart** at http://www.cs.nyu.edu/pipermail/fom/2007-July/011720.html: "On page 196 of that article, he gives a brief sketch of conversion to prenex normal form, remarking that it "can evidently be done."".

## 5.2. Skolem Normal Form

**Warning!** The principal results of this Section are valid only for the **classical logic**!

Skolem normal form was first introduced by Thoralf Skolem (1887-1963) in 1928:

**Th.Skolem.** Über die mathematische Logik. "Norsk matematisk tidsskrift", 1928, vol.10, pp.125-142.

The first very important idea was proposed by Skolem already in 1920:

**Th. Skolem.** Logisch-kombinatorische Untersuchungen über die Erfüllbarkeit und Beweisbarkeit mathematischen Sätze nebst einem Theoreme über dichte Mengen. *Videnskabsakademiet i Kristiania, Skrifter* I, No. 4, 1920, pp. 1-36.

Namely, according to Skolem's idea, further "normalization" becomes possible, if we drop the requirement that the "normal form" must be equivalent to the initial formula, and replace it by the requirement: **"normal form" must be logically valid; if and only if the initial formula is logically valid**. It appears, that in this way we can "reduce" any closed formula to a closed formula containing only one kind of quantifiers:

$$\exists x_1 \exists x_2 ... \exists x_n H(x_1, x_2, ..., x_n),$$

where H does not contain quantifiers at all (see Theorem 5.2.4 below).

Still, in his original formulation, instead of logical validity, Skolem was interested in a more technical notion – **satisfiability**. Let us remind that, in a predicate language L, a formula F is called satisfiable, if and only if there is an interpretation of the language L such that F is true for some values of its free variables. For our current purpose – **refutation proofs** (to prove that $F_1$, .., $F_n$ ⊢ G, we assume ¬G and try to derive a contradiction) satisfiability works as well as does logical validity. Indeed (verify, see [Exercise 4.1.1](#)), a set of formulas is inconsistent, if and only if it is unsatisfiable. Thus, if, in a refutation proof, we replace some formula H by an "equally satisfiable" formula H' (i.e. H' is satisfiable, if and only if so is H), then the refutation proof remains valid. Indeed, if, in this way, we derive a contradiction from $F_1$, .., $F_n$,¬G, then this set of formulas is, indeed, unsatisfiable, i.e. G logically follows from $F_1$, .., $F_n$ (for a more precise version of this argument see Exercise 5.2.4).

Skolem's second main idea (proposed in his 1928 paper): **allow introduction of new object constants and function constants**. It can be demonstrated on the following example: how could we "simplify" the formula $\forall x \exists y\, F(x, y)$? It asserts that for each x there is y such that F(x, y) is true. Thus, it asserts, that **there is a function** g, which selects for each value of x a value of y such that F(x, y) is true. Thus, in a sense, $\forall x \exists y\, F(x, y)$ is "equivalent" to $\forall x\, F(x, g(x))$. In which sense? In the sense that

$\forall x \exists y\, F(x, y)$ is satisfiable, if and only if $\forall x\, F(x, g(x))$ is satisfiable.

Indeed,

1. If $\forall x \exists y\, F(x, y)$ is satisfiable, then there is an interpretation J where it is true, i.e. for each value of x there is a value of y such that F(x, y) is true. This allows us to define the following interpretation of the function constant g: g(x) is one of y-s such that F(x, y) is true in J. If we extend J by adding this interpretation of the function constant g, we obtain an interpretation J', where $\forall x\, F(x, g(x))$ is true, i.e. this formula is satisfiable.

2. If $\forall x\, F(x, g(x))$ is satisfiable, then there is an interpretation J where it is true, i.e. for each value of x the formula F(x, g(x)) is true. Hence, in this interpretation, for each value of x there is a value of y (namely, g(x)) such that F(x, y) is true in J. Thus, $\forall x \exists y\, F(x, y)$ is true in J, i.e. this formula is satisfiable.

**Note**. In the first part of this proof, to define the function g, we need, in general, the Axiom of Choice. Indeed, if there is a non-empty set $Y_x$ of y-s such that F(x, y) is true, to define g(x), we must choose a single element of $Y_x$. If we know nothing else about the interpretation J, we are forced to use the Axiom of Choice. But, if we know that the interpretation J has a

countable domain, then we can define g(x) as the "least" y from the set $Y_x$. In this way we can avoid the Axiom of Choice.

The third idea is even simpler: the formula $\exists x\, F(x)$ asserts that there is x such that F(x) is true, so, let us denote by (an object constant) *c* one of these x-s, thus obtaining F(c) as a "normal form" of $\exists x\, F(x)$. Of course (verify),

$\exists x\, F(x)$ is satisfiable, if and only if F(c) is satisfiable.

These two ideas allow "reducing" of any quantifier prefix $Qx_1 Qx_2...Qx_n$ to a sequence of universal quantifiers only:

**Theorem 5.2.1 (Th. Skolem)**. Let L be a predicate language. There is an algorithm allowing to construct, for each *closed* formula F of this language, a *closed* formula F' (in a language L' obtained from L by adding a finite set of new object constants and new function constants – depending on F) such that:

a) F' is satisfiable, if and only if F is satisfiable,

b) F' is in form $\forall x_1 \forall x_2...\forall x_n G$, where n≥0, and G does not contain quantifiers.

If a formula is in form $\forall x_1 \forall x_2...\forall x_n G$, where n≥0, and G does not contain quantifiers, let us call it **Skolem normal form**. Thus, each closed formula can be reduced to a Skolem normal form in the following sense: for each closed formula F of a language L there is a Skolem normal form $|F|_{Sk}$ (in the language L extended by a finite set of Skolem constants and Skolem functions), which is satisfiable, if and only if so is F.

**Note**. In computer science slang, the reduction procedure leading to Skolem normal form is called "**skolemization**".

**Note.** Theorem 5.2.1 **does not** assert that a formula and its Skolem normal form are equivalent. It asserts only that the **satisfiability problem** of the first formula is equivalent to the satisfiability problem of the second formula. As already mentioned above, this is enough to allow using of Skolem reduction **in refutation proofs**.

Thus, if we are interested in determining the satisfiability of formulas, then reducing to Skolem normal forms is a promising method. Indeed, formulas $\forall x_1 \forall x_2...\forall x_n G$ (where G does not contain quantifiers) are, perhaps, easier to analyze than more complicated combinations of quantifiers.

**Proof of Theorem 5.2.1** First, let us obtain a prenex normal form $F_1$ of the formula F (see [Section 5.1](#)). Indeed, by Theorem 5.1.4, there is a simple algorithm, allowing to construct a closed formula $F_1$ such that $F_1$ is a prenex normal form, and, in the classical logic, $\vdash F \leftrightarrow F_1$. Of course, $F_1$ is satisfiable; if and only if so is F.

If the quantifier prefix of $F_1$ starts with a sequence of existential quantifiers ($\exists\exists...\exists\forall...$), we will need the following lemma to "reduce" these quantifiers:

**Lemma 5.2.2** . A closed formula $\exists x_1 \exists x_2 ... \exists x_n H(x_1, x_2, ..., x_n)$ is satisfiable, if and only if $H(c_1, c_2, ..., c_n)$ is satisfiable, where $c_1, c_2, ..., c_n$ are new object constants that do not appear in H.

After this operation, we have a closed prenex formula $H(c_1, c_2, ..., c_n)$ (in a language obtained from L by adding a finite set of new object constants, called **Skolem constants**), which is satisfiable, if and only if so is $F_1$ (and F). The the quantifier prefix of $H(c_1, c_2, ..., c_n)$ (if any) starts with a sequence of universal quantifiers ($\forall\forall...\forall\exists...$).

To proceed, we will need the following

**Lemma 5.2.3**. A closed formula $\forall x_1 \forall x_2 ... \forall x_n \exists y K(x_1, x_2, ..., x_n, y)$ is satisfiable, if and only if $\forall x_1 \forall x_2 ... \forall x_n K(x_1, x_2, ..., x_n, g(x_1, x_2, ..., x_n))$ is satisfiable, where g is a new n-ary function constant (called **Skolem function**), which does not appear in K.

By iterating this lemma, we can "reduce" the entire quantifier prefix of $H(c_1, c_2, ..., c_n)$ to a sequence of universal quantifiers only ($\forall\forall...\forall$).

For example, the formula $\exists t \forall x \forall y \exists z \forall u \exists w F(t,x,y,z,u,w)$ is satisfiable, if and only if so is

$$\forall x \forall y \exists z \forall u \exists w F(c,x,y,z,u,w)$$

(where $c$ is a constant that does not appear in F), and, if and only if so is

$$\forall x \forall y \forall u \exists w F(c,x,y,g(x,y),u,w) \ ,$$

and, if and only if so is the Skolem normal form:

$$\forall x \forall y \forall u F(c,x,y,g(x,y),u,h(x,y,u)) \ ,$$

where g and h are functions that do not appear in F.

**Exercise 5.2.1.** a) Prove Lemma 5.2.2. b) Prove Lemma 5.2.3.

How many new object constants and new function constants (Skolem constants and functions) do we need to obtain the final formula F'? The number of new symbols is determined by the number of existential quantifiers in the quantifier prefix of the prenex formula $F_1$. Indeed, a) the number of new object constants is determined by the number of existential quantifiers in front of the prefix, and b) the number of new function constants is determined by the number of existential quantifiers that follow after the universal ones.

This completes the proof of Theorem 5.2.1.

**Exercise 5.2.2**. Obtain Skolem normal forms of the formulas mentioned in Exercise 5.1.4.

See also:
"Skolemization" from The Wolfram Demonstrations Project. Contributed by: Hector Zenil.

Still, if we are interested in determining the logical validity of formulas, then we should apply the result of Exercise 4.1.1 together with Theorem 5.2.1:

F is logically valid, if and only if ¬F is not satisfiable, if and only if a Skolem normal form of ¬F is not satisfiable, if and only if $\forall x_1 \forall x_2 ... \forall x_n G$ (where n≥0, and G does not contain quantifiers) is not satisfiable, if and only if $\neg \forall x_1 \forall x_2 ... \forall x_n G$ is logically valid, if and only if $\exists x_1 \exists x_2 ... \exists x_n \neg G$ is logically valid.

Thus we have proved the following

**Theorem 5.2.4**. Let L be a first order language. There is an algorithm allowing to construct, for each closed formula F of this language, a closed formula F' (in a language L' obtained from L by adding a finite set of new object constants and new function constants – depending on F) such that:

a) F' is logically valid (or, provable in the classical logic), if and only if F is logically valid (or, provable in the classical logic),

b) F' is in form $\exists x_1 \exists x_2 ... \exists x_n G$, where n≥0, and G does not contain quantifiers.

**Exercise 5.2.3** (optional, for smart students). In his above-mentioned 1920 paper, for quantifier elimination, Skolem proposed **introduction of new predicate constants** (to the idea that function constants will do better, he arrived only in the 1928 paper). Do not read neither Skolem's papers, nor the above-mentioned online comments, and prove yourself that by introduction of new predicate constants, the satisfiability problem of any closed formula can be reduced to the satisfiability problem of a formula having the form $\forall x_1 \forall x_2 ... \forall x_m \exists y_1 \exists y_2 ... \exists y_n G$, where m, n≥0, and G does not contain quantifiers. Thus, function constants "will do better" – see Theorem 5.2.1.

**Exercise 5.2.4** (optional, compare with Exercise 5.1.5). Since, in general, Skolem normal form is not equivalent to the initial formula, we cannot use reduction to Skolem normal forms in the usual ("positive", or affirmative) proofs. But we may use it in "negative" (or, refutation) proofs, i.e. in proofs aimed at deriving a contradiction! More precisely, let us try extending the classical logic by introducing of the following additional inference rule (let us call it **SNF-rule**): given a formula F, replace it by some its Skolem normal form F' (such that the newly introduced object constants and function constants do not appear in the proof before F'). Verify, that, in fact, this rule does not extend the classical logic for refutation proofs, i.e. if, from a set of formulas $F_1$, $F_2$, ..., $F_n$, one can derive a contradiction by using [$L_1$-$L_{15}$, MP, Gen, SNF-rule], then one can do the same by using [$L_1$-$L_{15}$, MP, Gen]. (Thus, the SNF-rule is **admissible for refutation proofs** in the classical logic.)

## 5.3. Conjunctive and Disjunctive Normal Forms

**Warning!** The principal results of this Section are valid only for the **classical logic**!

Let us continue the "normalization" process that we started in and by reducing formulas to their Skolem normal forms, where universal quantifiers stand in front of a formula that does not contain quantifiers. How could we further "normalize" this "formula that does not contain quantifiers"?

**Step 1: eliminate equivalence**

First of all, we can eliminate all equivalence connectives because B↔C is only a shortcut for $(B \rightarrow C) \wedge (C \rightarrow B)$. Why should we? Because, proving of B↔C consists of proving of B→C and proving of C→B. Usually, using of the shortcut simplifies the appearance of the formula, not its proof.

**Step 2: eliminate implication**

After this, our formula will contain only implication, conjunction, disjunction and negation connectives. As the next step, we could try to eliminate one (or two?) of these connectives. The classical logic allows to do that. For example, by Theorem 2.6.4(b),

$[L_1\text{-}L_{11}, MP]: \vdash (A \rightarrow B) \leftrightarrow \neg A \vee B$ .

By using this equivalence, we can eliminate implication connectives. For example, the formula B→(C→D) is equivalent (in the classical logic only!) to $\neg B \vee (\neg C \vee D)$ .

But, instead of implications, we could try eliminating disjunctions or conjunctions as well. Indeed,

**Exercise 5.3.1**. In the classical logic $[L_1\text{-}L_{11}, MP]$, prove once more the following:

a) $\vdash (A \rightarrow B) \leftrightarrow \neg(A \wedge \neg B)$ .

b) $\vdash (A \vee B) \leftrightarrow (\neg A \rightarrow B)$ .

c) $\vdash (A \vee B) \leftrightarrow \neg(\neg A \vee \neg B)$ .

d) $\vdash (A \wedge B) \leftrightarrow \neg(A \rightarrow \neg B)$ .

e) $\vdash (A \wedge B) \leftrightarrow \neg(\neg A \vee \neg B)$ .

(For smart students) Determine, which parts of these equivalences can be proved in the constructive logic $[L_1\text{-}L_{10}, MP]$. End of Exercise 5.3.1.

By using these results, we could eliminate from our formulas any one (or any

two) of the three connectives – implication, conjunction, or disjunction.

However, the best decision would be eliminating only implications. Why? Because conjunction and disjunction are associative and commutative operations – and very much like addition and multiplication of numbers! For example, after reducing the formula B→(C→B) to $\neg B \vee (\neg C \vee B)$ , we can further transform it to $\neg B \vee \neg C \vee B$ and $(\neg B \vee B) \vee C$ – and conclude that it is "true and provable" (no surprise, it is Axiom $L_1$).

**Step 3: move negations down to atoms**

Thus, after Step 2, our formula contains only conjunction, disjunction and negation connectives. Now, let us remind the two de Morgan Laws:

Theorem 2.6.3, [$L_1$-$L_{11}$, MP]: $\vdash \neg(A \wedge B) \leftrightarrow \neg A \vee \neg B$ .

Theorem 2.4.9 (b), [$L_1$-$L_9$, MP] $\vdash \neg(A \vee B) \leftrightarrow \neg A \wedge \neg B$ .

By using these equivalencies, we can shift negations down – until the atoms of the formula. For example, let us transform the formula

$$((A \to B) \to C) \to B \wedge C \quad .$$

First, eliminate implications:

$\neg((A \to B) \to C) \vee (B \wedge C)$ ,
$\neg(\neg(A \to B) \vee C) \vee (B \wedge C)$ ,
$\neg(\neg(\neg A \vee B) \vee C) \vee (B \wedge C).$

Apply de Morgan Laws:

$$(\neg\neg(\neg A \vee B) \wedge \neg C) \vee (B \wedge C) \quad ,$$
$$(\neg(\neg\neg A \wedge \neg B) \wedge \neg C) \vee (B \wedge C) \quad ,$$
$$((\neg\neg\neg A \vee \neg\neg B) \wedge \neg C) \vee (B \wedge C) \quad .$$

Now, let us remind the Double Negation Law:

Theorem 2.6.1, [$L_1$-$L_{11}$, MP]: $\vdash \neg\neg A \leftrightarrow A$.

It allows dropping the excessive negations – we can replace $\neg\neg\neg A$ by $\neg A$ and $\neg\neg B$ – by B:

$$((\neg A \vee B) \wedge \neg C) \vee (B \wedge C) \quad .$$

**Note.** This form of formulas is called **negation normal form**. Namely, a formula is in negation normal form, if it is built of atoms with or without negations by using conjunctions and disjunctions only. I.e. a formula in negation normal form contains only conjunctions, disjunctions and negations, and negations are located at the atoms only. As we see, in the classical logic, any propositional formula can be reduced (is equivalent) to some formula in negation normal form.

Negation normal form is the starting point for an alternative (to the Resolution method described in this Section 5) method of automated theorem-proving – the so-called Method of Analytic Tableaux. One does not use skolemization here, one simply obtains the negation

normal form of the formula (with quantifiers inside) and after this, applies a specific tree algorithm of the Tableaux method. (My exposition for students, in Latvian: [Tablo algoritms](.).) **End of Note.**

**Step 4: two algebras**

After Step 3, our formula is built up by using:

a) atoms,

b) atoms preceded by a negation,

c) conjunction and disjunction connectives.

Conjunction and disjunction are associative and commutative operations. By the behavior of "truth values", conjunction is a kind of multiplication:

$$0 \wedge 0 = 0, 0 \wedge 1 = 1 \wedge 0 = 0, 1 \wedge 1 = 1 \quad,$$

and disjunction – a kind of addition:

$$0 \vee 0 = 0, 0 \vee 1 = 1 \vee 0 = 1, 1 \vee 1 = 1 \quad.$$

However, for these operations **two** distributive laws are valid (Theorem 2.3.3) – conjunction is distributive to disjunction, and disjunction is distributive to conjunction:

$[L_1\text{-}L_8, MP]: \vdash (A \wedge B) \vee C \leftrightarrow (A \vee C) \wedge (B \vee C) \quad,$
$[L_1\text{-}L_8, MP]: \vdash (A \vee B) \wedge C \leftrightarrow (A \wedge C) \vee (B \wedge C) \quad.$

Thus, both of the two decisions could be justified:

1) (Our first "algebra") Let us treat conjunction as multiplication and disjunction – as addition (+). Then the above formula

$((\neg A \vee B) \wedge \neg C) \vee (B \wedge C)$ takes the form ((A'+B)C')+BC (let us replace ¬A by the "more algebraic" A'). After this, the usual algebraic transformations yield the formula A'C'+BC'+BC. By transforming back to logic we would obtain: $(\neg A \wedge \neg C) \vee (B \wedge \neg C) \vee (B \wedge C) \quad.$

2) (Our second "algebra") Let us treat conjunction as addition (+) and disjunction – as multiplication. Then the above formula

$((\neg A \vee B) \wedge \neg C) \vee (B \wedge C)$ takes the form (A'B+C')(B+C). After this, the usual algebraic transformations yield the formula A'BB+A'BC+C'B+C'C. By transforming back to logic we would obtain:

$(\neg A \vee B \vee B) \wedge (\neg A \vee B \vee C) \wedge (\neg C \vee B) \wedge (\neg C \vee C) \quad.$

However, additional simplifying rules can be applied in these "algebras".

First rule – conjunction and disjunction are idempotent operations:

$[L_1\text{-} L_5, MP]: \vdash A \wedge A \leftrightarrow A$ (see [Section 2.2](.)).

$[L_1, L_2, L_5, L_6\text{-}L_8, MP]: \vdash A \lor A \leftrightarrow A$   (see Section 2.3).

Thus, in both of our "algebras": **A+A = AA = A**.

Second rule – $A \land \neg A$  (i.e. "false") is a kind of "zero" in the first "algebra", and a kind of "one" – in the second "algebra" (see Section 2.5):

$[L_1\text{-}L_{10}, MP]: \vdash B \lor (A \land \neg A) \leftrightarrow B$  ;

$[L_1\text{-}L_{10}, MP]: \vdash ((A \land \neg A) \land B) \lor C \leftrightarrow C$  .

Indeed, in the first "algebra", these formulas mean B+AA' = B and AA'B+C = C, i.e. we may think that AA'=0, B0=0, C+0=C. In the second "algebra", these formulas mean B(A+A') = B and (A+A'+B)C = C, i.e. we may think that A+A'=1, B1=B, C+1=1.

Third rule – $A \lor \neg A$  (i.e. "true") is a kind of "one" in the first "algebra", and a kind of "zero" – in the second "algebra"  (see Section 2.5):

$[L_1\text{-}L_{11}, MP]: \vdash B \land (A \lor \neg A) \leftrightarrow B$  ;

$[L_1\text{-}L_{11}, MP]: \vdash ((A \lor \neg A) \lor B) \land C \leftrightarrow C$  .

Indeed, in the first "algebra", these formulas mean B(A+A') = B and (A+A'+B)C = C, i.e. we may think that A+A'=1, B1=1, C+1=1. In the second "algebra". these formulas mean B+AA' = B and AA'B+C = C, i.e. we may think that AA'=0, B0=0, C+0=C.

Thus, **in both algebras,**

**AA'=0, B0=0, C+0=C, A+A'=1, B1=B, C+1=1.**

So, let us continue our example

1) (The first "algebra") The formula A'C'+BC'+BC is equivalent to A'C'+B(C'+C) = A'C'+B, or, if we return to logic: $(\neg A \land \neg C) \lor B$  . Such disjunctions consisting of conjunctions are called **disjunctive normal forms** (DNFs). In a DNF, each conjunction contains each atom no more than once – either without negation, or with it. Indeed, if it contains some atom X twice, then: a) replace XX by X, or b) replace X'X' by X', or c) replace XX' by 0 (in the latter case – drop the entire conjunction from the expression). In this way, for some formulas, we may obtain "zero", i.e. an **empty DNF**. Of course, such formulas take only false values ("false" is "zero" in the first "algebra"). And for some formulas, we may obtain "one", i.e. a kind of **"full" DNF**. Such formulas take only true values ("true" is "one" in the first "algebra").

2) (The second "algebra") The formula A'BB+A'BC+C'B+C'C is equivalent to A'B+A'BC+BC' = A'B(1+C)+BC' = A'B+BC', or, if we return to logic: $(\neg A \lor B) \land (B \lor \neg C)$  . Such conjunctions consisting of disjunctions are called **conjunctive normal forms** (CNFs). In a CNF, each disjunction

contains each atom no more than once – either without negation, or with it. Indeed, if it contains some atom X twice, then: a) replace XX by X, or b) replace X'X' by X', or c) replace XX' by 0 (in the latter case – drop the entire disjunction from the expression). In this way, for some formulas, we may obtain "zero", i.e. an **empty CNF**. Of course, such formulas take only true values ("true" is "zero" in the second "algebra"). And for some formulas, we may obtain "one", i.e. a kind of **"full" CNF**. Such formulas take only false values ("false" is "one" in the second "algebra").

Thus, we have proved the following

**Theorem 5.3.1.** In the classical logic, every propositional formula can be reduced to DNF and to CNF. More precisely, assume, the formula F has been built of formulas $B_1$, $B_2$, ..., $B_n$ by using propositional connectives only. Then:

a) There is a formula $F_1$, which is in a (possibly empty or full) disjunctive normal form over $B_1$, $B_2$, ..., $B_n$ such that $[L_1\text{-}L_{11}, MP]: \vdash F \leftrightarrow F_1$.

b) There is a formula $F_2$, which is in a (possibly empty or full) conjunctive normal form over $B_1$, $B_2$, ..., $B_n$ such that $[L_1\text{-}L_{11}, MP]: \vdash F \leftrightarrow F_2$.

**Exercise 5.3.2**. a) Build DNFs and CNFs of the following formulas. (Hint: the algorithm is explained in the above Steps 1-4.)

$$\neg(A \wedge B \to C) \quad,$$
$$(A \to B) \leftrightarrow (C \to D) \quad,$$
$$A \vee B \leftrightarrow C \vee D \quad,$$
$$A \wedge B \leftrightarrow C \wedge D \quad.$$

b) Build DNFs and CNFs of the following formulas:

$$\neg(A \vee \neg A) \quad,$$
$$((A \to B) \to A) \to A \quad,$$
$$(A \to B) \to ((\neg A \to B) \to B) \quad.$$

The notion of disjunctive normal form was known in 1885 to Oscar Howard Mitchell:

**Oscar Howard Mitchell**. On a New Algebra of Logic. In: *Studies in Logic by Members of the Johns Hopkins University,* 1885, pp. 72-106.

## 5.4. Clause Form

**Warning!** The principal results of this Section are valid only for the **classical logic**!

**Clause forms of propositional formulas**

Which form is more "natural" – DNF, or CNF? Of course, CNF is more natural. Indeed, a DNF $D_1 \vee D_2 \vee ... \vee D_m$ asserts that one (or more) of the formulas $D_i$ is true. This is a very complicated assertion – sometimes $D_1$ is true, sometimes $D_2$ is true, etc. But, if we have a CNF instead – $C_1 \wedge C_2 \wedge ... \wedge C_n$ ? It asserts that **all** the formulas $C_i$ are true, i.e. we can replace the long formula $C_1 \wedge C_2 \wedge ... \wedge C_n$ by a set of shorter formulas $C_1$, $C_2$, ..., $C_n$. For human reading and for computer processing, a set of shorter formulas is much more convenient than a single long formula.

Let us return to our example formula $((A \to B) \to C) \to B \wedge C$ of <u>Section 5.3</u>, for which we obtained a DNF $(\neg A \wedge \neg C) \vee B$ and a CNF:

$$(\neg A \vee B) \wedge (B \vee \neg C) \ .$$

Without a transformation, DNF may be hard for reading and understanding. The CNF is more convenient – it says simply that $\neg A \vee B$ is true and $B \vee \neg C$ is true.

As another step, making the formulas easier to understand, we could apply the following equivalences:

[$L_1$-$L_{11}$, MP]: $\vdash \ \neg A \vee B \leftrightarrow A \to B$ ,

[$L_1$-$L_{11}$, MP]: $\vdash \ \neg A \vee \neg B \vee C \leftrightarrow A \wedge B \to C$ ,

[$L_1$-$L_{11}$, MP]: $\vdash \ \neg A \vee B \vee C \leftrightarrow A \to B \vee C$ ,

[$L_1$-$L_{11}$, MP]: $\vdash \ \neg A \vee \neg B \vee C \vee D \leftrightarrow A \wedge B \to C \vee D$ ,

etc.

**Exercise 5.4.1.** Verify these equivalences by proving that, generally (in the classical logic),

[$L_1$-$L_{11}$, MP]: $\vdash \ \neg A_1 \vee \neg A_2 \vee ... \vee \neg A_m \vee B_1 \vee B_2 \vee ... \vee B_n \ \leftrightarrow$

$$(A_1 \wedge A_2 \wedge ... \wedge A_m \to B_1 \vee B_2 \vee ... \vee B_n) \ .$$

Thus, we can replace our set of two formulas $\neg A \vee B, B \vee \neg C$ by the set $A \to B, C \to B$ . The conjunction of these two formulas is equivalent to the initial formula $((A \to B) \to C) \to B \wedge C$ .

Formulas having the form

$$A_1 \wedge A_2 \wedge ... \wedge A_m \to B_1 \vee B_2 \vee ... \vee B_n \ ,$$

or, alternatively,

$$\neg A_1 \vee \neg A_2 \vee ... \vee \neg A_m \vee B_1 \vee B_2 \vee ... \vee B_n \quad ,$$

where $A_1, A_2, ... , A_m, B_1, B_2, ... , B_n$ are atoms, are called **clauses**. Clauses are well suited for computer processing. Indeed, in the computer memory, we can represent the above formula simply as a pair of sets of atoms – the negative set $\{A_1, A_2, ... , A_m\}$ and the positive set $\{B_1, B_2, ... , B_n\}$.

What, if one (or both) of these sets is (are) empty?

If, in the formula $\neg A_1 \vee \neg A_2 \vee ... \vee \neg A_m \vee B_1 \vee B_2 \vee ... \vee B_n$ , we have m = 0 and n > 0, then, of course, this formula asserts simply that $B_1 \vee B_2 \vee ... \vee B_n$ , i.e. "converting" it into an implication with empty premise

$$\rightarrow B_1 \vee B_2 \vee ... \vee B_n$$

leads us to the following definition: the clause $\rightarrow B_1 \vee B_2 \vee ... \vee B_n$ means the same as $B_1 \vee B_2 \vee ... \vee B_n$ .

If, in the formula $\neg A_1 \vee \neg A_2 \vee ... \vee \neg A_m \vee B_1 \vee B_2 \vee ... \vee B_n$ , we have m > 0 and n = 0, then, of course, this formula asserts simply that $\neg A_1 \vee \neg A_2 \vee ... \vee \neg A_m$ , i.e. "converting" it into an implication with empty consequence

$$A_1 \wedge A_2 \wedge ... \wedge A_m \rightarrow$$

leads us to the following definition: the clause $A_1 \wedge A_2 \wedge ... \wedge A_m \rightarrow$ means the same as $\neg(A_1 \wedge A_2 \wedge ... \wedge A_m)$ .

If m=n=0, then, as an empty disjunction, the clause must be qualified as false.

**Note.** Clauses are similar to sequents – pairs of sets of formulas $(S_1, S_2)$, used in the proof of Theorem 4.4.5 (completeness of the constructive propositional logic) in <u>Section 4.4</u>. In a sequent $(S_1, S_2)$, the sets $S_1$, $S_2$ could contain arbitrary formulas, but, in a clause, $S_1$, $S_2$ are sets of atoms.

Sets (i.e. conjunctions) of clauses are called **clause forms** (in some texts – *clausal forms*). By Theorem 5.3.1, every propositional formula can be reduced to a (possibly empty, i.e. true) CNF. Since every conjunction member of a CNF represents, in fact, a clause, we have established the following

**Theorem 5.4.1**. In the classical logic, every propositional formula can be reduced to an equivalent clause form. More precisely, assume, the formula F is built of formulas $B_1$, $B_2$, ..., $B_n$ by using propositional connectives only. Then there is a (possibly empty) clause form F' (i.e. a set of clauses) over $B_1$, $B_2$, ..., $B_n$ such that $[L_1$-$L_{11}$, MP]: F $\leftrightarrow$ conj(F'), where conj(F') denotes the

conjunction of the clauses contained in the set F'.

For example, as we established above, the set $\neg A \vee B, B \vee \neg C$ (or, alternatively, $A \rightarrow B, C \rightarrow B$ ) is a clause form of the formula $((A \rightarrow B) \rightarrow C) \rightarrow B \wedge C$ .

**Exercise 5.4.2**. Obtain clause forms of the formulas mentioned in the Exercise 5.3.2.

Clause forms (in a sense, "clouds of simple disjunctions") are well suited for computer processing. In the computer memory, every clause

$$\neg A_1 \vee \neg A_2 \vee ... \vee \neg A_m \vee B_1 \vee B_2 \vee ... \vee B_n$$

can be represented as a pair of sets of atoms:

$$(-\{A_1, A_2, ..., A_m\}, +\{B_1, B_2, ..., B_n\}),$$

and every clause form – as a set of such pairs – i.e. it means less character string processing and less expression parsing!

**Clause forms of predicate formulas**

Of course (unfortunately), if we would insist that the clause form must be equivalent to the initial formula, then nothing comparable to clause forms could be obtained for predicate formulas. Still, reducing of predicate formulas to "clause forms" becomes possible, if we drop this requirement, and replace it by the requirement that the clause form must be satisfiable, if and only if the initial formula is satisfiable (or, equivalently, if we wish only to retain the possibility of deriving contradictions). And – if we allow the Skolem style extending of the language by adding new object constants and new function constants.

Then, by Skolem's Theorem (Theorem 5.2.1), for each **closed** formula F, we can obtain a **Skolem normal form** $\forall x_1 \forall x_2 ... \forall x_k G$, where k≥0, the formula G does not contain quantifiers, and this form is satisfiable, if and only if so is F.

As the next step, let us **drop the universal quantifiers** in front of G. In this way the satisfiability is retained: G is satisfiable, if and only if so is $\forall x_1 \forall x_2 ... \forall x_k G$. (Remember that G, as a formula containing free variables, is considered as true in an interpretation, if and only if it is true, in this interpretation, for all values of its free variables.)

**Exercise 5.4.3**. There is another way how to verify this. Use the *Gen*-rule and the axiom $L_{12}$ to verify that G allows deriving of contradictions, if and only if $\forall x_1 \forall x_2 ... \forall x_k$ G allows it.

Thus, if we drop the universal quantifiers, then nothing is lost. The possibility of deriving contradictions is retained.

As the next step, by Theorem 5.3.1, let us convert G into a **CNF**, and then – into a clause form G', i.e into a set of clauses G' = {$C_1$, $C_2$, ..., $C_m$} (with **atomic** sub-formulas of G playing the role of atoms $B_1$, $B_2$, ..., $B_n$). Since conj(G') is equivalent to G, conj(G') is satisfiable, if and only if so is F. Thus, the possibility of deriving contradictions is retained, again.

However, one more step is necessary to separate clauses completely – renaming of variables in such a way that **no two clauses contain common variables**.

Remember that we obtained our clauses from a closed formula in Skolem normal form: $\forall x_1 \forall x_2 ... \forall x_k G$. Since conj(G') is equivalent to G, $\forall x_1 \forall x_2 ... \forall x_k$ conj(G') is equivalent to $\forall x_1 \forall x_2 ... \forall x_k G$. But $\forall x_1 \forall x_2 ... \forall x_k$ conj(G') is equivalent to

$$(\forall x_1 \forall x_2 ... \forall x_k C_1) \wedge ... \wedge (\forall x_1 \forall x_2 ... \forall x_k C_m) .$$

According to the Replacement Theorem 3, we will obtain an equivalent formula, if we will rename the variables $x_i$ in such a way that no two clauses contain common variables.

Let us denote by G'' = {$C'_1$, $C'_2$, ..., $C'_m$} the set of separated clauses obtained via renaming of variables. The set G'' is called a **clause form** of the formula F. Remember that clauses are disjunctions of **atomic** formulas (with, or without negation), i.e. disjunctions of formulas having the form $p(t_1, ..., t_m)$ , or $\neg p(t_1, ..., t_m)$ , where p is a predicate constant, and $t_1$, ..., $t_m$ are terms (containing variables and/or object constants).

Thus, we have proven the following

**Theorem 5.4.2**. Let L be a predicate language. There is an algorithm allowing to construct, for each *closed* formula F of this language, a clause form S, i.e. a finite set of clauses (in a language L' obtained from L by adding a finite set of new object constants and new function constants – depending on F, and no two closes containing common variables) such that the set S is satisfiable, if and only if so is F.

As an example, let us consider the formula asserting that there are infinitely many prime numbers:

$$prime(x) \; : \; x > 1 \wedge \neg \exists y \exists z (y > 1 \wedge z > 1 \wedge x = y * z) ,$$

$$\forall u \exists x (x > u \wedge prime(x)) ,$$

$$\forall u \exists x (x > u \wedge x > 1 \wedge \neg \exists y \exists z (y > 1 \wedge z > 1 \wedge x = y * z)) \qquad (1)$$

First , let us convert it into a prenex normal form:

$$\forall u \, \exists x \, (x > u \wedge x > 1 \wedge \forall y \, \forall z \, \neg(y > 1 \wedge z > 1 \wedge x = y * z)) \quad ,$$

$$\forall u \, \exists x \, \forall y \, \forall z \, (x > u \wedge x > 1 \wedge \neg(y > 1 \wedge z > 1 \wedge x = y * z)) \quad .$$

After this, let us eliminate $\exists x$ by introducing a Skolem function x=g(u):

$$\forall u \, \forall y \, \forall z \, (g(u) > u \wedge g(u) > 1 \wedge \neg(y > 1 \wedge z > 1 \wedge g(u) = y * z)) \quad .$$

This is a Skolem normal form, let us drop the universal quantifiers:

$$g(u) > u \wedge g(u) > 1 \wedge (\neg(y > 1) \vee \neg(z > 1) \vee \neg(g(u) = y * z)) \quad ,$$

and let us convert this quantifier-free formula into a conjunctive normal form:

$$g(u) > u \wedge g(u) > 1 \wedge (\neg(y > 1) \vee \neg(z > 1) \vee \neg(g(u) = y * z)) \quad .$$

By dropping the conjunctions we obtain 3 clauses:

$$g(u) > u \quad , \quad g(u) > 1 \quad , \quad \neg(y > 1) \vee \neg(z > 1) \vee \neg(g(u) = y * z) \quad .$$

As the last step, we must rename the variables in such a way that no two clauses contain common variables:

$$g(u_1) > u_1 \quad ,$$
$$g(u_2) > 1 \quad ,$$
$$\neg(y > 1) \vee \neg(z > 1) \vee \neg(g(u_3) = y * z) \quad .$$

or, alternatively,

$$\rightarrow g(u_1) > u_1 \quad ,$$
$$\rightarrow g(u_2) > 1 \quad ,$$
$$y > 1, z > 1, g(u_3) = y * z \rightarrow \quad .$$

These sets of 3 formulas are clause forms of the formula (1). **They are satisfiable, if and only if so is the initial formula (1).**

**Exercise 5.4.4**. Obtain clause forms of the formulas mentioned in the Exercise 5.1.4 (assume that B, C, D, F are predicate constants).


**Clause form of a set of formulas**

Knowledge bases are, as a rule, large sets of closed formulas $F_1$, $F_2$, ..., $F_n$, i.e., in fact, large conjunctions $F_1 \wedge F_2 \wedge ... \wedge F_n$ of closed formulas. The clause form of this set can be obtained, simply as the **union** of clause forms of separate formulas $F_i$. However, each formula must be "kept separated" during the entire process:

a) when reducing to Skolem normal forms, the name of each new Skolem

constant and Skolem function must be chosen as "completely new" with respect to the entire process;

b) when renaming clause variables, at the end, we must guarantee that no two clauses of the entire process contain common variables.

**Horn clauses**

If, in a clause

$$A_1 \wedge A_2 \wedge ... \wedge A_m \rightarrow B_1 \vee B_2 \vee ... \vee B_n \quad ,$$

or, alternatively,

$$\neg A_1 \vee \neg A_2 \vee ... \vee \neg A_m \vee B_1 \vee B_2 \vee ... \vee B_n \quad ,$$

we have n=1 or n=0, then it is called Horn clause. I.e.,

$$A_1 \wedge A_2 \wedge ... \wedge A_m \rightarrow B \quad ,$$

or, alternatively,

$$\neg A_1 \vee \neg A_2 \vee ... \vee \neg A_m \vee B \quad .$$

There are formulas that cannot be reduced to Horn clauses (verify).

The name "Horn Clause" comes from the logician Alfred Horn, who first pointed out the significance of such clauses in 1951:

**A. Horn**. "On sentences which are true of direct unions of algebras", Journal of Symbolic Logic, 16, 14-21.

## 5.5. Resolution Method for Propositional Formulas

**Warning!** The principal results of this Section are valid only for the **classical logic**!

Remember, that we are solving here the problem of determining, does the formula G follow from the formulas $F_1$, $F_2$, ..., $F_n$. It does so, if and only if the set of formulas $F_1$, $F_2$, ..., $F_n$, ¬G is unsatisfiable. Assume, we have obtained a clause form S of the formula $F_1 \wedge F_2 \wedge ... \wedge F_n \wedge \neg G$ . Then S is unsatisfiable, if and only if so is the set $F_1$, $F_2$, ..., $F_n$, ¬G. How to determine, is S unsatisfiable, or not? In a sense, S represents "a cloud of simple disjunctions". How to work with such a cloud effectively?

## History

**J. A. Robinson.** Theorem-proving on the computer. "Jour. Assoc. Comput. Mach.", vol.10, N2, 1963, pp.163-174

**J. A. Robinson.** A machine-oriented logic based on the resolution principle, "Jour. Assoc. Comput. Mach.", vol.12, N1, January 1965, pp.23-41 (available online, Russian translation available: "Kib. sbornik (novaya seriya)", 7, 1970, pp.194-218)

John Alan Robinson: "Born in Yorkshire in 1930, Robinson came to the United States in 1952 with a classics degree from Cambridge University. He studied philosophy at the University of Oregon before moving to Princeton where he received his PhD in philosophy in 1956. Temporarily ``disillusioned with philosophy,`` he went to work as an operations research analyst for Du Pont, where he learnt programming and taught himself mathematics. Robinson moved to Rice University in 1961, spending his summers as a visiting researcher at the Argonne National Laboratory's Applied Mathematics Division. Its then Director, William F. Miller, pointed Robinson in the direction of theorem proving...

Miller showed Robinson a 1960 paper by Martin Davis and Hilary Putnam (coincidentally, the latter had been Robinson's PhD supervisor) proposing a predicate-calculus proof procedure that seemed potentially superior to Gilmore's, but which they had not yet turned into a practical computer program. Miller suggested that Robinson use his programming skills to implement Davis and Putnam's procedure on the Argonne IBM 704. Robinson quickly found that their procedure remained very inefficient. However, while implementing a different procedure also suggested in 1960 by Dag Prawitz, Robinson came to see how the two sets of ideas could be combined into a new, far more efficient, automated proof procedure for first-order predicate logic: "resolution"..." (According to Donald MacKenzie, The Automation of Proof: A Historical and Sociological Exploration, "IEEE Annals of the History of Computing", vol.17, N3, 1995, pp. 7-29).

**Note.** Almost at the same time when J.A.Robinson invented the resolution method, Sergei Maslov invented his **inverse method**, which has a similar range of applications:

**S. Yu. Maslov**. An inverse method of establishing deducibilities in the classical predicate calculus, "Soviet Mathematics, Doklady"*, 1964, N5, pp.1420-1423*.

See also: Maslov S. Y. (1939-1982), human rights activist in ENCYCLOPAEDIA OF SAINT PETERSBURG.

About the history of the problem see:

**J. A. Robinson.** Computational Logic: Memories of the Past and Challenges for the Future. *Computational Logic – CL 2000, First International Conference, London, UK, 24-28 July, 2000, Proceedings*, Springer, Lecture Notes in Computer Science, 2000, Vol. 1861, pp. 1-24 (online copy).

**M. Davis**. The Early History of Automated Deduction.In: *Handbook of Automated Reasoning*, ed. by A. Robinson and A. Voronkov, Elsevier Science, 2001, vol. I, pp. 3-15.

## The Method

Again, how to work with "a cloud of simple disjunctions" effectively?

Assume that, in a set of clauses, two clauses are contained such that an atom C appears as a positive member in the first clause, and as a negative member in

the second one:

$$\neg A_1 \vee \neg A_2 \vee ... \vee \neg A_m \vee B_1 \vee B_2 \vee ... \vee B_n \vee C \quad , \qquad (1)$$

$$\neg C \vee \neg D_1 \vee \neg D_2 \vee ... \vee \neg D_p \vee E_1 \vee E_2 \vee ... \vee E_q \quad , \qquad (2)$$

or, simply,

$$F \vee C \quad , \qquad (1a)$$

$$\neg C \vee G \quad . \qquad (2a)$$

If C is false, then (1a) yields F, and, if C is true, then (2a) yields G. Thus, from (1a) and (2a) we have derived $F \vee G$ . I.e. deriving of $F \vee G$ from $F \vee C$ and $\neg C \vee G$ is "logically correct", and it is called **Robinson's Resolution rule** (J. A. Robinson proposed it in the above 1963 paper):

$$\frac{F \vee C , \neg C \vee G}{F \vee G} \quad .$$

Taking into account the rule (of the classical logic) $\neg A \vee B \leftrightarrow (A \rightarrow B)$ , we can obtain an alternative form of the Resolution rule:

$$\frac{\neg F \rightarrow C , C \rightarrow G}{\neg F \rightarrow G} \quad .$$

In the classical logic, this form is equivalent to the **Law of Syllogism** (transitivity of implication).

If F is empty, then this form derives G from C, C→G, i.e. **Resolution rule includes Modus Ponens** as a special case.

If G is empty, then from $\neg F \vee C , \neg C$ (i.e. F→C, ¬C), the **Resolution rule** derives ¬F, i.e. it **includes Modus Tollens** as a special case.

**Exercise 5.5.1**. Derive the Resolution rule in the constructive logic, i.e. prove that [$L_1$-$L_{10}$, MP]: $C \vee F , \neg C \vee G \vdash F \vee G$ . Verify that it cannot be proved in the minimal logic [$L_1$-$L_9$, MP]. (Hint: in the positive part – use Theorem 2.5.1(b) [$L_1$, $L_2$, $L_8$, $L_{10}$, MP]: $F \vee C , \neg C \vdash F$ . In the negative part – verify that in the minimal logic, the Resolution rule allows proving of $L_{10}$, see Section 2.5).

Thus, from the clauses (1) and (2), Robinson's Resolution rule allows deriving of the following clause:

$$\neg A_1 \vee \neg A_2 \vee ... \vee \neg A_m \vee \neg D_1 \vee \neg D_2 \vee ... \vee \neg D_p \vee B_1 \vee B_2 \vee ... \vee B_n \vee E_1 \vee E_2 \vee ... \vee E_q$$

At first glance, this approach leads to nothing, because this formula seems to be much longer than (1), and than (2). Still, this is not 100% true, because, additionally, we can reduce the repeating atoms, and, finally, the set of

different atoms, used in a clause form, is fixed! If, in our set of clauses, there are N different atoms, then none of the clauses (initial, or generated by resolution) will contain more than N atoms (each with or without negation). And the total number of different clauses will never exceed $3^N$ (*missing, without negation, with negation*). Thus, repeated applications of the Resolution rule will "rotate" within this restricted "search space".

The smart idea behind Robinson's Resolution rule is as follows: it is a universal tool for **deriving contradictions from inconsistent sets of clauses**! No other axioms and rules of inference are necessary! More precisely, it is universal, if used together with the following trivial rules of inference:

$$\frac{F \vee \boldsymbol{C} \vee \boldsymbol{D} \vee G}{F \vee \boldsymbol{D} \vee \boldsymbol{C} \vee G} \text{ (permutation),}$$

$$\frac{F \vee \boldsymbol{C} \vee \boldsymbol{C} \vee G}{F \vee \boldsymbol{C} \vee G} \text{ (reduction).}$$

The permutation rule allows arbitrary reordering of atoms in a clause (for example, moving C to right, and moving ¬C to left). The reduction rule allows reduction of repeating identical atoms.

**Exercise 5.5.2** (optional). Derive these inference rules in the minimal logic [$L_1$-$L_9$, MP].

**Theorem 5.5.1 (J. A. Robinson)**. In the classical propositional logic [$L_1$-$L_{11}$, MP], a finite set of propositional clauses is inconsistent, if and only if Robinson's Resolution rule (together with permutation and reduction rules) allows for deriving of a contradiction from it.

**Note**. In some other texts, this fact is called "the refutation-completeness of the Resolution rule" for the propositional logic.

**Proof**. 1. As you have proved in the Exercises 5.5.1 and 5.5.2, all the formulas, derived from a set of formulas $K_1$, $K_2$, ... , $K_s$ by using the Permutation, Resolution and Reduction rules are consequences of $K_1$, $K_2$, ... , $K_s$. Hence, if these rules allow deriving a contradiction from this set of formulas, then it (the set) is inconsistent.

2. Now, let us assume that a set of propositional clauses $K_1$, $K_2$, ... , $K_s$ is inconsistent, i.e. a contradiction $A \wedge \neg A$ can be derived from it:

$$[L_1\text{-}L_{11}, \text{MP}]: K_1, K_2, ... , K_s \vdash A \wedge \neg A \ .$$

Then, under the classical truth tables, the conjunction $K_1 \wedge K_2 \wedge ... \wedge K_s$ takes only false values (verify!). Let us mark one of the atoms (the atom C) in it. Let us denote:

- by $C \vee F_i$ – the clauses containing C without negation,

- by $\neg C \vee G_j$ – the clauses containing C with negation,

- by $H_k$ – the clauses that do not contain C.

All the formulas $F_i$, $G_j$, $H_k$ are disjunctions of atoms (with or without negations) that do not contain the atom C.

Thus $K_1 \wedge K_2 \wedge ... \wedge K_s$ is equivalent to

$$conj(C \vee F_i) \wedge conj(\neg C \vee G_j) \wedge conj(H_k) \quad . \qquad (4)$$

Let us apply (the strange) one of the distribution rules (Theorem 2.3.1):

$$[L_1\text{-}L_8, MP] \vdash (A \wedge B) \vee C \leftrightarrow (A \vee C) \wedge (B \vee C) \quad .$$

Hence, $K_1 \wedge K_2 \wedge ... \wedge K_s$ is equivalent to

$$(C \vee conj(F_i)) \wedge (\neg C \vee conj(G_j)) \wedge conj(H_k) \quad .$$

If C is false, then this formula is equivalent to $conj(F_i) \wedge conj(H_k)$, i.e. $conj(F_i) \wedge conj(H_k)$ takes only false values. If C is true, then it is equivalent to $conj(G_j) \wedge conj(H_k)$, i.e. $conj(G_j) \wedge conj(H_k)$ takes only false values. Thus the disjunction

$$(conj(F_i) \wedge conj(H_k)) \vee (conj(G_j) \wedge conj(H_k)) \qquad (5)$$

also takes only false values. Now, let us, apply (the "normal") one of the distribution rules (Theorem 2.3.1):

$$[L_1\text{-}L_8, MP] \vdash (A \vee B) \wedge C \leftrightarrow (A \wedge C) \vee (B \wedge C) \quad ,$$

obtaining that (5) is equivalent to

$$(conj(F_i) \vee conj(G_j)) \wedge conj(H_k) \quad . \qquad (6)$$

I.e. this formula also takes only false values. And – important note! – it does not contain the atom C.

Finally, by applying, again, (the strange) one of the distribution rules (Theorem 2.3.1) we can conclude that (6) is equivalent to $conj(conj(F_i \vee G_j)) \wedge conj(H_k)$, i.e. to the set of clauses $F_i \vee G_j$ and $H_k$ (where i, j, k run over their initial ranges).

What does this achievement mean? If the set of propositional clauses $K_1$, $K_2$, ... , $K_s$ is inconsistent, then there is a set of clauses $F_i \vee G_j$ and $H_k$ (where i, j, k run over their initial ranges), which is inconsistent as well, but which contains one atom less than $K_1$, $K_2$, ... , $K_s$.

Now, imagine, that, in the clause form (4), we have applied the Resolution rule for the atom C **in all the possible ways** (before applying, apply the

permutation rule to reorder atoms moving C to right, and ¬C – to left):

$$\frac{F_i \vee C, \neg C \vee G_j}{F_i \vee G_j} \quad .$$

After this, apply the permutation and reduction rules to reduce identical atoms. In this way we have obtained exactly the above-mentioned inconsistent set of clauses $F_i \vee G_j$ and $H_k$ (where i, j, k run over their initial ranges).

Thus, if some set of propositional formulas $K_1, K_2, \ldots, K_s$ is inconsistent, then the Resolution rule (togeher with the permutation and reduction rules) allows to derive from it another inconsistent set of propositional formulas, which contains one atom less.

By iterating this process, at the end of it, we will have an inconsistent set of propositional formulas built of a single atom B. In a clause form, there can be only one such set – the set B, ¬B. This set represents a contradiction.

Q.E.D.

As an example, let us use Robinson's Resolution rule to prove that

$$B \vee C, C \rightarrow B, B \rightarrow D \;\vdash\; B \wedge D \quad .$$

Let us add $\neg(B \wedge D)$ to the premises $B \vee C, C \rightarrow B, B \rightarrow D$ . We must prove that this set of 4 formulas is inconsistent. First, let us obtain clause forms:

$B \vee C$ in clause form is $B \vee C$ ,

$C \rightarrow B$ in clause form is $\neg C \vee B$ ,

$B \rightarrow D$ in clause form is $\neg B \vee D$ ,

$\neg(B \wedge D)$ is equivalent to $\neg B \vee \neg D$ .

Now, let us apply resolution to derive a contradiction from this set of 4 clauses: BvC, ¬CvB, ¬BvD, ¬Bv¬D:

From $B \vee \boldsymbol{C}, \neg \boldsymbol{C} \vee B$ we derive B, and have now 5 clauses:

$$B \vee C, \neg C \vee B, \neg B \vee D, \neg B \vee \neg D, B \quad .$$

From $\neg B \vee \boldsymbol{D}, \neg B \vee \neg \boldsymbol{D}$ we derive ¬B, and have now 6 clauses:

$$B \vee C, \neg C \vee B, \neg B \vee D, \neg B \vee \neg D, B, \neg B \quad .$$

We have derived a contradiction: B, ¬B. This proves that the formula $B \wedge D$ follows from $B \vee C, C \rightarrow B, B \rightarrow D$ . Q.E.D.

**Exercise 5.5.3.** Use the Resolution rule to prove the following:

a) A→B, ¬A→B ⊢ B.

b) (A→B)→A ⊢A (Peirce's Law).

c) B→(C→D), B→C ⊢B→D (Axiom $L_2$).

d) B→D, C→D ⊢ $B \lor C \to D$ . (Axiom $L_8$).

e) $A \lor B \lor C$, $B \to A \lor C$, $A \to C$ ⊢C.

## From a programmer's point of view

Of course, when implementing the Resolution rule in a computer program, we do not need decorations like the permutation and reduction rules. In a program, we will represent each clause $\neg A_1 \lor \neg A_2 \lor ... \lor \neg A_m \lor B_1 \lor B_2 \lor ... \lor B_n$ as a pair of sets: negative atoms, N = {$A_1$, $A_2$, ... , $A_m$}, and positive atoms, P = {$B_1$, $B_2$, ... , $B_n$}. Of course, the sets N, P do not intersect (if they do, then this clause contains $\neg C \lor C \lor ...$ , i.e. it can be dropped as "non-informative").

**Resolution rule (non-refined version).** If there are two clauses $N_1$, $P_1$ and $N_2$, $P_2$ such that $P_1$ and $N_2$ (or $N_1$ and $P_2$) contain a common atom C, then we can derive the clause $N_1 \cup N_2 - \{C\}, P_1 \cup P_2 - \{C\}$ .

Of course, the set union operation includes reduction of identical members automatically.

The condition "$P_1$ and $N_2$ (or $N_1$ and $P_2$) contain a common atom C" can be expressed as $C \in (P_1 \cap N_2) \cup (P_2 \cap N_1)$ .

If, in the resulting clause, the sets $N_1 \cup N_2 - \{C\}, P_1 \cup P_2 - \{C\}$ intersect, then we should ignore such result. Fortunately, this can be detected in advance. Indeed,

$$(N_1 \cup N_2) \cap (P_1 \cup P_2) = (N_1 \cap P_1) \cup (N_1 \cap P_2) \cup (N_2 \cap P_1) \cup (N_2 \cap P_2) =$$
$$(P_1 \cap N_2) \cup (P_2 \cap N_1) ,$$

because $N_1 \cap P_1$, $N_2 \cap P_2$ are empty sets. The set $(P_1 \cap N_2) \cup (P_2 \cap N_1)$ is exactly the set of all atoms C allowing application of the Resolution rule to clauses $N_1$, $P_1$ and $N_2$, $P_2$. Hence, the sets $N_1 \cup N_2 - \{C\}, P_1 \cup P_2 - \{C\}$ will not intersect, if and only if the set $(P_1 \cap N_2) \cup (P_2 \cap N_1)$ **contains exactly one atom** C, i.e., if and only if there is exactly one atom allowing application of the Resolution rule.

**Resolution rule (refined version).** If there are two clauses $N_1$, $P_1$ and $N_2$, $P_2$ such that the set $(P_1 \cap N_2) \cup (P_2 \cap N_1)$ contains exactly one atom C, then we can derive the clause $N_1 \cup N_2 - \{C\}, P_1 \cup P_2 - \{C\}$ .

Now, let us try to design a program implementing the last step of "proving by

resolution" – suppose, we have already the **initial list** of clauses, and we wish to apply the Resolution rule trying to derive a contradiction.

The main data storage will be a growing list of clauses (the **main list**):

$$(N_1, P_1), (N_2, P_2), ..., (N_k, P_k), ...$$

It will start as the initial list, and each application of the Resolution rule will append a new clause to it.

To guarantee a success, we must apply the Resolution rule **in all the possible ways**, i.e. we must scan all pairs of clauses $(N_i, P_i)(N_j, P_j)$, where i = 1, 2, ...; j = i+1, i+2, ... To achieve this, let us use the following pair enumeration process:

$(N_1, P_1)(N_2, P_2)$ – first, scan all pairs (i, j) with **j=2**, i<j.

$(N_1, P_1)(N_3, P_3)$, $(N_2, P_2)(N_3, P_3)$ – after this, scan all pairs (i, j) with **j=3**, i<j.

$(N_1, P_1)(N_4, P_4)$, $(N_2, P_2)(N_4, P_4)$, $(N_3, P_3)(N_4, P_4)$ – after this, scan all pairs (i, j) with **j=4**, i<j.

Etc.

The process will stop, when we will arrive at the level j, and the main list will contain less than j (in fact, j-1) clauses. For a set of n atoms, there are only $3^n$ different clauses. For example, for two atoms A, B, there are 9 different clauses: $\neg A \vee \neg B, \neg A \vee B, A \vee \neg B, A \vee B, \neg A, A, \neg B, B$ , and the empty clause (representing contradiction). I.e., if we will prohibit duplicate clauses in the main list, then our process will always stop.

Thus, the following pseudo-code will do (no string processing, no expression parsing necessary!):

```
function propositional resolution (initial list) { of clauses }
begin
if initial list contains contradiction then return TRUE { contradiction found }
main list = eliminate duplicates (initial list)
for j = 2 by 1
begin
- if count (main list) < j then return FALSE { no contradiction derived }
- else
- for i = 1 to j−1 by 1
-- { consider i-th and j-th clauses in the main list: (Ni, Pi), (Nj, Pj) }
```

-- if $(N_i \cap P_j) \cup (P_i \cap N_j)$ contains exactly one element C **then**
```
-- begin
--- {apply resolution}
```

--- **if** ( $N_i \cup N_j - \{C\}, P_i \cup P_j - \{C\}$ ) **not in** main list **then**
--- **begin**
---- **add** it **to** main list
---- **if** main list contains contradiction **then return** TRUE { contradiction derived }
-- **end**
--- **end**
**end**
**end**

**Exercise 5.5.4** (optional). Develop a computer program implementing the above pseudocode.

**Note**. See my version of such a program in C++: header file, implementation).

## Warning!

Despite its beauty, resolution method cannot overcome the general complexity problem, mentioned at the end of Section 4.2: in the classical propositional logic, the task of reasoning is "co-NP-complete". And a closer analysis shows that, in the **worst possible case**, given a set of formulas of total length *n,* the time required by resolution method will be exponential – about $2^{Cn}$ seconds. But in many **practical situations**, experience shows that resolution method solves its task, and – in acceptable time. In particular, Prolog interpreters are using resolution, and are solving many practical tasks in acceptable time!

## 5.6. Herbrand's Theorem

**Warning!** The principal results of this Section are valid only for the **classical logic**!

Jacques Herbrand (1908-1931) "... After leaving Göttingen, Herbrand decided on a holiday in the Alps before his intended return to France. However he was never to complete his plans for he died in a mountaineering accident in the Alps only a few days after his holiday began. His death at the age of 23 in one of the tragic losses to mathematics." (according to MacTutor History of Mathematics archive).

Herbrand proved his famous theorem in 1929:

**J.Herbrand.** Recherches sur la théorie de la démonstration. Ph.D. Thesis, University of Paris, 1930 (approved in April 1929).

Unlike the proof presented below, the original proof of Herbrand's Theorem does not depend on Gödel's Completeness Theorem (or Model Existence Theorem). Herbrand completed his Ph.D. thesis in 1929. In the same 1929 Gödel completed his doctoral dissertation about completeness (see Section 4.3). In fact, Herbrand's method allows proving of Gödel's Completeness Theorem, but he (Herbrand) "did not notice it". Why? See

The flavour of this theorem can be best presented in its simplest version. In this version, F(x) is a quantifier-free formula containing only one variable x. Then, Herbrand's Theorem says:

The formula $\exists x F(x)$ is logically valid, if and only if there is a finite set of constant terms $t_1$, ..., $t_n$ such that the disjunction $F(t_1) \vee ... \vee F(t_n)$ is logically valid.

Or, equivalently (via Gödel's Completeness Theorem),

The formula $\exists x F(x)$ is provable in the classical logic, if and only if there is a finite set of constant terms $t_1$, ..., $t_n$ such that the disjunction $F(t_1) \vee ... \vee F(t_n)$ is provable in the classical logic.

As we will see in the proof, Herbrand's Theorem is "caused" by the simple "fact" that in any proof of $\exists x F(x)$ only a finite set of terms can be used.

Now, more precisely:

Let L be a predicate language, containing at least one object constant, and let F be a quantifier-free formula.

Idea #1 (author?). The formula $p(c_1) \wedge q(c_2, f(x))$ is quantifier-free ($c_1$, $c_2$ are object constants, f – a function constant, p, q – predicate constants). In a sense, any "closed" interpretation domain for this formula must contain objects denoted by the terms $c_1$, $c_2$, $f(c_1)$, $f(c_2)$, $f(f(c_1))$, $f(f(c_2))$,...

So, let us define the so-called **Herbrand's universe** of the formula F (let us denote it by $HU_F$) as the minimum set of all constant terms such that:

a) If c is an object constant occurring in F, then c is in $HU_F$.

b) If F does not contain object constants, then one of the constants of the language L is in $HU_F$.

c) If terms $t_1$, ..., $t_k$ are in $HU_F$, and f is a k-ary function constant occurring in F, then the term $f(t_1, ..., t_k)$ is in $HU_F$.

**Exercise 5.6.1.** Verify that $HU_F$ is a non-empty finite or countable set (provide an algorithm generating the members of $HU_F$).

**Theorem 5.6.1 (**Herbrand's Theorem – the simplest case). Let L be a predicate language, containing at least one object constant, and let F(x) be a quantifier-free formula containing only one free variable x. Then the formula $\exists x F(x)$ is **logically valid (i.e. provable in the classical predicate logic)**, if and only if

there is a finite set of terms $t_1$, ..., $t_n$ from $HU_F$ such that the disjunction $F(t_1) \vee ... \vee F(t_n)$ is **logically valid (i.e. provable in the classical predicate logic)**.

**Proof.** Let us assume the contrary – that none of the disjunctions $F(t_1) \vee ... \vee F(t_n)$ is logically valid ($t_i$-s are terms from $HU_F$). Idea #2 – then the following theory T is consistent:

$$T = \{ \neg F(t) \mid t \text{ is a term from } HU_F \}.$$

Indeed, if T would be inconsistent, then there would be a T-proof of some formula B&¬B. In this proof, only a finite set of the axioms ¬F(t) would be used, i.e. for some terms $t_1$, ..., $t_n$ from $HU_F$:

$$[L_1\text{-}L_{15}, MP, Gen]: \neg F(t_1), ..., \neg F(t_n) \vdash B \wedge \neg B \quad.$$

Hence, by Deduction Theorem 2 (it is applicable here, because F(x) contains only one free variable, and $t_i$-s are constant terms, i.e. every ¬F($t_i$) is a closed formula):

$$[L_1\text{-}L_{15}, MP, Gen]: \vdash \neg F(t_1) \wedge ... \wedge \neg F(t_n) \rightarrow B \wedge \neg B \quad,$$

$$[L_1\text{-}L_{15}, MP, Gen]: \vdash \neg(F(t_1) \vee ... \vee F(t_n)) \rightarrow B \wedge \neg B \quad,$$

and thus,

$$[L_1\text{-}L_{15}, MP, Gen]: \vdash F(t_1) \vee ... \vee F(t_n) \quad.$$

I.e., $F(t_1) \vee ... \vee F(t_n)$ is logically valid. This contradicts our assumption, that none of the disjunctions $F(t_1) \vee ... \vee F(t_n)$ is logically valid. Hence, T is a consistent theory.

Idea #3 – if T is consistent, then, by the Model Existence Theorem, there is a model J of T. In this model, all the axioms of T are true, i.e. so are all the formulas ¬F(t) with t from the set $HU_F$.

Idea #4 – let us restrict the domain of the model J to those elements of it, which are interpretations of terms from the set $HU_F$, and let us restrict the entire interpretation correspondingly. Let us denote this new interpretation by $J_1$. Then,

a) All the formulas ¬F(t) (with t from the set $HU_F$) are true in $J_1$. Indeed, ¬F(t) contains only constant terms from $HU_F$ (idea #1 working!), and all of them have the same interpretations in $J_1$ that they had in J. Thus, if ¬F(t) was true in J, it remains true in $J_1$.

b) Hence, the formula $\forall x \neg F(x)$ is true in $J_1$ (because the domain of $J_1$ consists only of those elements, which are interpretations of terms from the set $HU_F$).

c) Hence, the formula $\exists x F(x)$ is false in $J_1$.

This contradicts the logical validity of $\exists x F(x)$.

Q.E.D.

**Exercise 5.6.2** (optional). Repeat the above proof, proving a more general form of Herbrand's Theorem:

**Theorem 5.6.2** (Herbrand's Theorem – the simplest case). Let L be a predicate language, containing at least one object constant, and let $F(x_1, ..., x_m)$ be a quantifier-free formula containing only m free variables $x_1, ..., x_m$. The formula $\exists x_1...\exists x_m F(x_1, ..., x_m)$ is **logically valid**, if and only if there is a finite set of m-tuples $tt_1, ..., tt_n$ of terms from $HU_F$ such that the disjunction $F(tt_1) \vee ... \vee F(tt_n)$ is **logically valid**.

As you verified it in the Exercise 4.1.1, any formula G is logically valid, if and only if $\neg G$ is unsatisfiable. Thus, $\exists x_1...\exists x_m F(x_1, ..., x_m)$ is logically valid, if and only if $\forall x_1...\forall x_m \neg F(x_1, ..., x_m)$ is unsatisfiable. On the other hand, $F(tt_1) \vee ... \vee F(tt_n)$ is logically valid, if and only if $\neg F(tt_1) \wedge ... \wedge \neg F(tt_n)$ is unsatisfiable. Now, let us replace F by $\neg$F, and we have proved

**Theorem 5.6.3** (Herbrand's Theorem – a more useful alternative form). Let L be a predicate language, containing at least one object constant, and let $F(x_1, ..., x_m)$ be a quantifier-free formula containing only m free variables $x_1, ..., x_m$. The formula $\forall x_1...\forall x_m F(x_1, ..., x_m)$ is **unsatisfiable** (i.e. **inconsistent** in the classical logic), if and only if there is a finite set of m-tuples $tt_1, ..., tt_n$ of terms from $HU_F$ such that the conjunction $\neg F(tt_1) \wedge ... \wedge \neg F(tt_n)$ is **unsatisfiable** (i.e. **inconsistent** in the classical logic).

**Note.** As you verified it in the Exercise 4.3.7, a set of formulas is inconsistent in the classical logic, if and only if it is unsatisfiable.

Why is this form "more useful"? Let us try applying this form of Herbrand's Theorem to sets of formulas in clause form.

1) The "meaning" of any set of closed formulas $F_1, ... , F_k$ is represented by their conjunction $F_1 \wedge ... \wedge F_k$ .

2) A clause is any disjunction of atomic formulas or their negations. For

example, $\neg p(c_1) \lor p(c_2) \lor q(x, f(y))$ , or $p(x) \lor \neg q(y, f(z))$ . The "meaning" of a set of clauses is represented by their universally quantified conjunction. For example,

$$\forall x \, \forall y \, \forall z ([\neg p(c_1) \lor p(c_2) \lor q(x, f(y))] \land [p(x) \lor \neg q(y, f(z))]) \ .$$

3) As we know from the previous <u>Section 5.4</u>, the set $F_1, \dots F_k$ can be reduced to a clause from, i.e. there is a set of clauses S such that $F_1, \dots, F_k$ is unsatisfiable, if and only if S is unsatisfiable.

Now, let us apply the above form of Herbrand's Theorem (Theorem 5.6.3). If S contains m variables (of course, all of them are universally quantified), then S is unsatisfiable, if and only if there is a finite set of m-tuples $tt_1, \dots, tt_n$ of terms from $HU_S$ such that the conjunction $S(tt_1) \land \dots \land S(tt_n)$ is unsatisfiable.

If we take a clause from S, and substitute some terms from $HU_S$ for all its variables, then we obtain a (so-called) **ground clause** of S. For example, if

$$S = \{ \ \neg p(c_1) \lor p(c_2) \lor q(x, f(y)) \ ; \ p(x) \lor \neg q(y, f(z)) \ \},$$

then the substitution $\{ c_1 / x; c_2 / y; f(c_2) / z \}$ yields the following two ground clauses:

$$\neg p(c_1) \lor p(c_2) \lor q(f(c_1), f(c_2)) \ ,$$
$$p(c_1) \lor \neg q(c_2, f(f(c_2))) \ .$$

Of course, the conjunction $S(tt_1) \land \dots \land S(tt_n)$ is a set of ground clauses. Thus, **if S is unsatisfiable, then there is an unsatisfiable finite set of ground clauses of S.** And conversely?

If there is an unsatisfiable finite set $C = \{C_1, \dots, C_n\}$ of ground clauses of S, then each $C_i$ is generated by some substitution, which can be represented as an m-tuple $tt_i$ of terms from $HU_S$. If $\{C_1, \dots, C_n\}$ is unsatisfiable, then $\{S(tt_1), \dots, S(tt_n)\}$ – as a super-set of the former, is unsatisfiable, too ("even more unsatisfiable").

Now, if S would be satisfiable, then (because all the variables of S are meant universally quantified) so would be the formula $S(tt_1) \land \dots \land S(tt_n)$ . Contradiction.

Thus, we have proved another form of Herbrand's Theorem.

**Theorem 5.6.4** (Herbrand's Theorem – the most useful form. Author – <u>Herbert B.Enderton</u>?). Let L be a predicate language, containing at least one object constant, and let $F_1, \dots, F_k$ be a set of closed formulas in L. Then this set is

**unsatisfiable**, if and only if its clause form allows an **unsatisfiable** finite set of ground clauses.

Why is this form "the most useful"? Because (let us ignore performance problems),

a) The clause form of $F_1$, ..., $F_k$ is a finite set S, generated by a simple (but a very slow) algorithm (see Sections 5.1-5.4).

b) Herbrand's universe $HU_S$ is a finite or infinite set of constant terms, generated by a simple algorithm (see Exercise 5.6.1).

c) Thus, all the possible finite sets of ground clauses of S can be generated by a simple combination of the above two algorithms.

d) Unsatisfiability of each finite set of ground clauses can be detected by a simple (but a very slow) algorithm (see Lemma 5.6.5 below).

Thus, we have here a **simple algorithm** (but a very slow one) **for checking provability in the classical predicate logic.**

**Lemma 5.6.5**. A finite set of ground clauses is unsatisfiable, if and only if the conjunction of these clauses is unsatisfiable under the classical truth tables.

**Proof**. In the above example of ground clauses:

$$\neg p(c_1) \lor p(c_2) \lor q(f(c_1), f(c_2)) \quad,$$
$$p(c_1) \lor \neg q(c_2, f(f(c_2))) \quad,$$

we have 5 different atoms: $p(c_1)$, $p(c_2)$, $q(f(c_1), f(c_2))$, $q(c_2, f(f(c_2)))$. Let us denote these atoms by $Q_1$, $Q_2$, $Q_3$, $Q_4$. Thus we obtain the following propositional formula

$$(\neg Q_1 \lor Q_2 \lor Q_3) \land (Q_1 \lor \neg Q_4) \quad.$$

1. If this formula **cannot** be satisfied under the classical truth tables, then we cannot assign truth values to predicates p, q in a way making all the corresponding clauses true. I.e. then the corresponding set of ground clauses also cannot be satisfied. Q.E.D.

2. If this formula **can** be satisfied under the classical truth tables, then we can find a truth value assignement making it true, for example:

$Q_1$=false (this makes the first disjunction true),

$Q_4$=false (this makes the second disjunction true).

Now, we can define the following interpretation J making the corresponding ground clauses true:

$D_J$ = { $c_1$, $c_2$, $f(c_1)$, $f(c_2)$, $f(f(c_2))$ } (the set of all terms appearing in the clauses, i.e. a subset of the Herbrand universe);

$p(c_1)$=false, $q(c_2, f(f(c_2)))$=false (these assignements make both ground clauses true).

All the other truth values are irrelevant, so, we can define them, for example, as follows:

$p(c_2)$=true, $p(f(c_1))$=true, $p(f(c_2))$=true, $p(f(f(c_2)))$=true;

$q(x, y)$=true, if x is not $c_2$, or y is not $f(f(c_2))$.

Q.E.D.

...

To be continued.

...

## 5.7. Resolution Method for Predicate Formulas

**Warning!** The principal results of this Section are valid only for the **classical logic**!

If we are interested only in deriving contradictions from inconsistent sets of formulas, then we can note that a set of closed predicate formulas is inconsistent (i.e. allows deriving a contradiction in the classical logic), if and only if the conjunction of these formulas is unsatisfiable (Exercise 4.3.7). Thus, instead of the initial set, we can analyze the set of clause forms of these formulas. Indeed, if we derive a contradiction from the set of clause forms, then this set is unsatisfiable, i.e., by Theorem 5.4.2, so is the initial set, and hence, the initial set is inconsistent. And conversely, if the initial set of formulas is consistent, then it is satisfiable, i.e. so is the set of clause forms, i.e. we will be not able to derive a contradiction from it.

The next step forward – in clause forms, we can **drop all the universal quantifiers**. Indeed, if we derive a contradiction from a set universally quantified clause forms, then we can derive it from the corresponding non-quantified set (we can apply the Gen inference rule $F(x) \vdash \forall x F(x)$ to obtain the quantified forms from the non-quantified ones). And conversely, if we derive a contradiction from a set of non-quantified clause forms, then we can derive it from the corresponding universally quantified set (apply the Axiom $L_{12}$:

∀xF(x) → F(x) to obtain non-quantified forms from the quantified ones).

After dropping quantifiers, sets of clause forms become simply sets of clauses (conjunction of conjunctions is equivalent to a "joint" conjunction).

Thus, we can concentrate on sets of clauses that do not contain quantifiers, like as the one obtained in Section 5.4:

$$\rightarrow g(u_1) > u_1 \quad,$$
$$\rightarrow g(u_2) > 1 \quad,$$
$$y > 1, z > 1, g(u_3) = y * z \rightarrow \quad.$$

Note that clauses consist of **atomic** formulas only, and **no two clauses contain common variables**. Thus, clauses are completely separated, and this separation will greatly simplify processing of clauses by means of substitution (see below).

Will the Robinson's Resolution rule remain a universal tool for deriving contradictions also from inconsistent sets of predicate formulas (i.e. sets of non-quantified clauses, consisting of atomic formulas)?

Let us imagine, we have derived the following two formulas (p is a unary predicate constant, 0 – an object constant):

$$p(x_1) \vee F(x_1, y_1) \quad, \quad \neg p(0) \vee G(x_2, y_2) \quad.$$

To apply the Robinson's Resolution rule, we must first, in $p(x_1)$, substitute 0 for $x_1$:

$$p(0) \vee F(0, y_1) \quad, \quad \neg p(0) \vee G(x_2, y_2) \quad.$$

Now, we can apply the Resolution rule, obtaining the formula

$$F(0, y_1) \vee G(x_2, y_2) \quad.$$

Surprisingly, this simple idea of "**unification by substitution**" is sufficient to make Robinson's Resolution rule a universal tool for deriving contradictions also from inconsistent sets of predicate formulas! And, in general, the necessary substitutions are not much more complicated than in the above simplest example.

**Note**. In fact, unification is a very general phenomenon in human and computer reasoning – it appears as one of the main components in deductive, inductive and analogical reasoning as well. More – at the end of this Section.

The **substitution rule** allows, in some clause C, replacing of all occurrences of some variable x by any term t.

**Theorem 5.7.1 (J. A. Robinson)**. In the classical predicate logic [$L_1$-$L_{11}$, $L_{12}$-$L_{15}$, MP, Gen], a set of predicate clauses is inconsistent, if and only if

Robinson's **Resolution rule** (together with permutation, reduction and **substitution** rules) allows deriving a contradiction from it.

**Note**. In some other texts, this fact is called "the refutation-completeness of the Resolution rule".

**Proof**. 1. All the formulas, derived from a set of clauses $K_1$, $K_2$, ... , $K_s$ by using permutation, reduction, substitution and Resolution rules, are consequences of $K_1$, $K_2$, ... , $K_s$. Hence, if these rules allow deriving a contradiction from this set of clauses, then it (the set) is inconsistent.

2. Now, let us assume that the set of clauses S = $\{K_1$, $K_2$, ... , $K_s\}$ is inconsistent. Then it is unsatisfiable ([Exercise 4.3.7](#)). And then, by Herbrand's Theorem, it allows a finite unsatisfiable set of ground clauses $C_1$, ..., $C_n$. Each $C_i$ of these ground clauses is obtained from some clause in S by means of some substitution $sub_i$ (of terms from the Herbrand universe $HU_S$), i.e. by applying the substitution rule.

By Lemma 5.6.5, the set $C_1$, ..., $C_n$ is unsatisfiable, if and only if the conjunction $C_1 \land ... \land C_n$ is unsatisfiable under the classical truth tables, i.e.**, if and only if** the set $C_1$, ..., $C_n$ is inconsistent. And, by Theorem 5.5.1, a finite set of propositional clauses is inconsistent; if and only if Robinson's Resolution rule (together with permutation and reduction rules) allows deriving a contradiction from it.

Q.E.D.

### Refinements – Step 1 (first of the two smart ideas)

Let us examine once more the part two of the proof of Theorem 5.7.1, where a specific (hopeless!) "proof strategy" is used.

First, since two clauses $K_i$ do not contain common variables, we can think that each of the substitutions $sub_j$ is applied to a single clause, i.e. we can think, in fact, of a (finite) set of substitutions $sub_{ij}$, where each $sub_{ij}$ is applied only to the clause $K_i$. Let us denote by F.sub the result of application of the substitution sub to the formula F.

Second, to derive a contradiction from $\{K_1$, $K_2$, ... , $K_s\}$, we may apply, **first**, all the necessary substitutions (stage 1 – substitutions only!), and, **after this**, all the necessary permutations, reductions and resolutions (stage 2 – no more substitutions!). This is exactly the above-mentioned specific (hopeless!) "proof strategy". Why hopeless? Because, before applying the substitutions $sub_{ij}$, we must **find them among all the possible substitutions** of terms from the

infinite set $HU_S$. This is a performance problem that does not affect our above theoretical considerations, but could make their result useless. The smart ideas #1 and #2 introduced below, allow to restrict the substitution search space considerably.

Imagine one of the resolutions of stage 2, where $C_1$ is an atomic formula:

$$\frac{F_1 \vee C_1, \neg C_1 \vee G_1}{F_1 \vee G_1} \quad .$$

If both premises $F_1 \vee C_1, \neg C_1 \vee G_1$ are coming directly from stage 1, then they have been obtained from some initial clauses $F \vee C, \neg D \vee G$ by two substitutions $sub_1$ and $sub_2$ such that:

$$F_1 \text{ is } F.sub_1, \ C_1 \text{ is } C.sub_1, \ \neg C_1 \text{ is } \neg D.sub_2, \ G_1 \text{ is } G.sub_2.$$

We can call such pair of substitutions a **unifier**, because $C.sub_1$ and $D.sub_2$ represent the same atomic formula (compare the example before the text of Theorem 5.7.1).

If one (or both) of the premises does not come directly from stage 1, then it is either an initial clause, or the result of a previous resolution. By putting an empty substitution (which does no change formulas) instead of $sub_1$ or $sub_2$ (or both) we can still think of the premises as obtained by a unification.

And, finally, if, to derive a contradiction B, ¬B from $K_1, K_2, \ldots, K_s$, we do not need resolution at all, then we need, nevertheless, unifying substitutions, converting two clauses $B_1$ and $\neg B_2$ into B and ¬B.

Thus (smart idea #1), **to derive contradictions, we can do with one specific kind of the substitution rule – the unification rule**:

a) Take two clauses, mark a positive atom C in the first clause, and a negative atom ¬D in the second one. Thus, we are considering two clauses: $F \vee C$ and $\neg D \vee G$ .

b) Try to find two substitutions $sub_1$ and $sub_2$ such that $C.sub_1$ and $D.sub_2$ represent the same atom $C_1$. And you do not need to introduce variables of the other clauses! If you succeed, you have obtained two clauses: $F_1 \vee C_1, \neg C_1 \vee G_1$ , where $C_1$ is $C.sub_1$ (=$D.sub_2$), $F_1$ is $F.sub_1$ and $G_1$ is $G.sub_2$. Since clauses do not contain common variables, the union $sub_1 \cup sub_2$ is a substitution (a unifier of C and D).

c) Apply resolution, obtaining the clause $F_1 \vee G_1$ .

We have proved the following refined version of Theorem 5.7.1:

**Theorem 5.7.2 ( J.A.Robinson)**. In the classical predicate logic [$L_1$-$L_{11}$, $L_{12}$-$L_{15}$, MP, Gen], a set of predicate clauses is inconsistent; if and only if Robinson's **Resolution rule** (together with permutation, reduction and **unification** rules) allows deriving a contradiction from it.

Why is this refinement important? Because now, instead of trying out all the possible substitutions (of terms from $HU_S$ for clause variables), we can concentrate on substitutions that unify two clauses. This allows to restrict the substitution search space considerably.

**Refinements – Step 2 (second of the two smart ideas)**

**Substitution "algebra"**

In general, each substitution involves a list of distinct variables $x_1$, ..., $x_k$ and a list of terms $t_1$, ...,$t_k$. All occurrences of the variable $x_i$ are replaced by the term $t_i$. Thus, this operation can be most naturally represented by the set of pairs { $t_1$ / $x_1$, ..., $t_k$ / $x_k$ }. The order of pairs $t_i$ / $x_i$ is irrelevant because of the following "anti-cascading" condition: the new occurrences of the variables $x_1$, ..., $x_k$ created by the substitution, are not replaced. The result of application of some substitution sub to some expression (term or formula) F, is usually denoted by F.sub.

For example, if F is p(x, f(y)) and sub = { f(z) / x, z / y }, then F.sub is p(f(z), f(z)).

The empty set of pairs {} represents the so-called empty substitution. Of course, F.{} = F, for any expression F.

If the variable sets of two substitutions $sub_1$ and $sub_2$ do not intersect, and the terms of $sub_1$ do not contain the variables of $sub_2$, and the terms of $sub_2$ do not contain the variables of $sub_1$, then the **union** $sub_1 \cup sub_2$ (of two sets of pairs) defines a substitution.

Still, the most important operation on substitutions is **composition**. If $sub_1$ and $sub_2$ are two substitutions, then $sub_1.sub_2$ denotes the composed substitution "apply first $sub_1$, and after this, apply $sub_2$". For example, if $sub_1$ = { f(z) / x, z / y } and $sub_2$ = { f(w) / z }, then

$$sub_1.sub_2 = \{ \ f(f(w)) \ / \ x, \ f(w) \ / \ y, \ f(w) \ / \ z \ \}.$$

**Exercise 5.7.2** (optional). a) Verify that the substitution composition is associative and non-commutative (provide a counter-example), and that the empty substitution is the only "unit

element" (i.e. {}.sub = sub.{} = sub for any substitution sub). b) Is there any algebraic correlation between composition and union of substitutions?

**Most general unifiers**

How do behave unifiers in the substitution "algebra"? Assume, $sub_1$ and $sub_2$ are two different unifiers of the same pair of expressions F and G. I.e.

$$F.sub_1 = G.sub_1, \quad F.sub_2 = G.sub_2.$$

If there would be a substitution sub such that $sub_2 = sub_1.sub$, then we could say that $sub_1$ is a **no less general unifier** than $sub_2$. For example, let us try to unify the first members of the following two formulas:

$$p(x_1) \vee F(x_1, y_1) \quad , \quad \neg p(f(x_2)) \vee G(x_2, y_2) \quad .$$

It would be natural to use the substitution $sub_1 = \{ f(z) / x_1, z / x_2 \}$, obtaining

$$p(f(z)) \vee F(f(z), y_1) \quad , \quad \neg p(f(z)) \vee G(z, y_2) \quad .$$

But, in principle, one could use also the substitution $sub_2 = \{ f(f(z)) / x_1, f(z) / x_2 \}$, obtaining

$$p(f(f(z))) \vee F(f(f(z)), y_1) \quad , \quad \neg p(f(f(z))) \vee G(f(z), y_2) \quad .$$

Of course, $sub_1$ is "better", because $sub_2 = sub_1.\{ f(z) / z \}$. Why? If our purpose was unifying $p(x_1)$ with $p(f(x_2))$, then $sub_1$ performs this (as well as $sub_2$), but it "leaves more space" for subsequent substitutions (than $sub_2$). Indeed, to continue after $sub_1$, instead of $sub_2 = sub_1.\{ f(z) / z \}$, we can choose also $sub_3 = sub_1.\{ g(z) / z \}$ etc. Thus, using a more general unifier is preferable.

So, let us call a unifier sub of two expressions F and G a **most general unifier (mgu)** of F and G, if and only if it is no less general than any other unifier of F and G (i.e. if and only if, for any other unifier sub' of F and G, there is a substitution sub" such that sub' = sub.sub").

**Lemma 5.7.3.** If two expressions lists FF and GG are unifiable, then there exists an mgu of FF and GG.

**Proof** (long, but easy). Let us define the **total length** of an expression list as follows: a) (atomic expressions) the total length of a constant or of a variable is 1, b) the total length of the expression list $e_1, ..., e_n$ is the sum of the total lengths of the members $e_1, ..., e_n$, c) (composite expressions) the total length of the expression $f(t_1, ..., t_n)$ (where f is function constant or predicate constant), is the total length of the expression list $t_1, ..., t_n$ plus 1.

Let us prove our Lemma by induction using

$$\min(\text{total\_length}(FF), \text{total\_length}(GG))$$

as the induction parameter.

1) **Induction base.** The total length of FF or GG is 1. Let us assume $\text{total\_length}(FF)=1$.

a) FF is a constant c. Then FF and GG are unifiable, if and only if GG is the same constant c. Then, empty substitution is the only possible mgu (verify).

b) FF is a variable x. Then, FF and GG are not unifiable, if: $b_1$) GG is a list of more than one expression, or, $b_2$) GG is a composite expression that contains x (then any substitution of t for x makes GG longer than t). And, FF and GG are unifiable, if and only if GG is a variable, or GG is a composite expression that does not contain x.

If GG is the variable x, then the empty substitution is the only possible mgu (verify).

If GG is a variable y (other than x), then all unifications of FF and GG have the form { t / x, t / y, ... }, where t is any term. Among them, mgu-s are { z / x, z / y }, where z is any variable (verify).

If GG is a composite expression that does not contain x, then all unifications of FF and GG have the form { GG.sub / x, ... } U sub, where sub is any substitution that does not substitute for x (verify). Among them, mgu-s are {GG.sub / x}U sub, where sub substitutes distinct variables for distinct variables (verify).

This completes the induction base.

2) **Induction step.** Assume, $\min(\text{total\_length}(FF), \text{total\_length}(GG))=n$, where $n>1$. If FF and GG are unifiable, then, as lists, they contain the same number of members.

2a) FF and GG contain are single expressions. Since $\min(\text{total\_length}(FF), \text{total\_length}(GG))>1$, both are composite expressions – suppose, FF is $f(s_1, ..., s_m)$ (where f is function constant or predicate constant, and $s_1, ..., s_m$ are terms), and GG is $g(t_1, ..., t_n)$ (where g is function constant or predicate constant, and $t_1, ..., t_n$ are terms). FF and GG are unifiable, if and only if a) f and g represent the same constant, and b) the lists $s_1, ..., s_m$ and $t_1, ..., t_n$ are unifiable. Thus, the unifiers of FF and GG coincide with the unifiers of lists. Since $\min(\text{total\_length}(s_1, ..., s_m), \text{total\_length}(t_1, ..., t_n))<n$, by the induction assumption, Lemma 5.7.3 holds for the lists, i.e. it holds also for FF and GG.

2b) FF and GG contain two or more members. If FF and GG are unifiable, then so are their first members ("heads") $F_1$ and $G_1$. Let us denote by $FF_2$ and $GG_2$ the rests of lists ("tails"). Since min(total_length($F_1$), total_length($G_1$))<n, by the induction assumption, there exists at least one mgu of $F_1$ and $G_1$. The same is true also for $FF_2$ and $GG_2$.

Let us denote by $mgu_1$ an arbitrary mgu of $F_1$ and $G_1$

Now, let us consider an arbitrary unifier u of FF and GG. It must unify also $F_1$ with $G_1$, and $FF_2$ with $GG_2$. Hence, $u = mgu_1.sub_1$, where $sub_1$ is some substitution. We know that $F_1.mgu_1 = G_1.mgu_1$.

But what about $FF_2.mgu_1$ and $GG_2.mgu_1$? Let us apply $sub_1$ to both:

$$FF_2.mgu_1.sub_1 = FF_2.u$$

$$GG_2.mgu_1.sub_1 = GG_2.u$$

Since u unifies $FF_2$ with $GG_2$,

$$FF_2.mgu_1.sub_1 = GG_2.mgu_1.sub_1,$$

i.e. $sub_1$ unifies $FF_2.mgu_1$ with $GG_2.mgu_1$. Let us denote by $mgu_{12}$ an arbitrary mgu of $FF_2.mgu_1$ and $GG_2.mgu_1$. Then, $sub_1 = mgu_{12}.sub_{12}$, where $sub_{12}$ is some substitution, and

$$mgu_1.mgu_{12}.sub_{12} = mgu_1.sub_1 = u.$$

Thus, we have established that for an arbitrary unifier u of FF and GG there is a substitution $sub_{12}$ such that $mgu_1.mgu_{12}.sub_{12} = u$. Of course, the composition $mgu_1.mgu_{12}$ unifies FF with GG (since it unifies $F_1$ with $G_1$, and $FF_2$ with $GG_2$). Hence, $mgu_1.mgu_{12}$ is an mgu of FF and GG.

Q.E.D.

**Unification algorithm**

How could we determine, can two atomic formulas C and D be unified, or not? This problem can be solved by the following simple pseudo-code *GetMostGeneralUnifier*, which follows the above proof of Lemma 5.7.3, and where **expression lists** are defined in the LISP style:

1) Each variable, constant, function constant and predicate constant is an expression list (consisting of a single member).

2) If $s_1$, ..., $s_n$ are expression lists, then the list of $s_1$, ..., $s_n$ is an expression list (consisting of members $s_1$, ..., $s_n$). The first member $s_1$ is called the **head** of the list, and the list of $s_2$, ..., $s_n$ – the **tail** of the list.

Thus, instead of, for example, $f(t_1, .., t_n)$, we use simply the (LISP style) list f, $t_1$, .., $t_n$. This simplifies the recursion interface.

This program detects, are two expression lists unifiable, or not, and, if they are, it returns one of their most general unifiers.

**function** GetMostGeneralUnifier (expression_list1, expression_list2)
**begin**
**if** length(expression_list1) > 1 **and** length(expression_list2) > 1 **then**
**begin**
--- h1 = head(expression_list1);
--- h2 = head(expression_list2);
--- subH = GetMostGeneralUnifier(h1, h2);
--- **if** subH = false **then return** false; {unification impossible}
--- t1 = tail(expression_list1).subH;
--- t2 = tail(expression_list2).subH;
--- subT = GetMostGeneralUnifier(t1, t2);
--- **if** subT = false **then return** false; {unification impossible, note that subH is a mgu!}
--- **return** subH.SubT; {this composition unifies expression_list1 and expression_list2}
**end**
{now, expression_list1, or expression_list2 consists of a single member: m1 or m2}
**if** length(expression_list1) = 1 **and** m1 is variable **then**
**begin**
--- **if** m1 = expression_list2 **then return** {}; {empty substitution}
--- **if** m1 occurs in expression_list2 **then return** false; {unification impossible – verify!}
--- **return** {expression_list2 / m1}; {substitute expression_list2 for m1}
**end**
**if** length(expression_list2) = 1 **and** m2 is variable **then**
**begin**
--- **if** m2 = expression_list1 **then return** {}; {empty substitution}
--- **if** m2 occurs in expression_list1 **then return** false; {unification impossible – verify!}
--- **return** {expression_list1 / m2}; {substitute expression_list1 for m2}
**end**
{now, expression_list1, or expression_list2 consists of a single member that is

not variable}
if expression_list1 = expression_list2 **then return** {}; {empty substitution}
**return** false; {unification impossible – verify!}
**end**

**Exercise 5.7.3.** Verify that this program detects, are two expression lists unifiable, or not, and, if they are, it returns one of their mgu-s. (Hint: repeat the proof of Lemma 5.7.3.)

**Smart idea #2:**

**To derive contradictions, we can do with even more specific kind of the unification rule – the mgu-rule**:

a) Take two clauses, mark a positive atom C in the first clause, and a negative atom ¬D in the second one. Thus, we are considering two clauses: FvC and ¬DvG.

b) Try to find any mgu of C and D. If you succeed, you have obtained two clauses: $F.mgu \vee C_1, \neg C_1 \vee G.mgu$ , where $C_1$ is C.mgu (=D.mgu).

c) Apply resolution, obtaining the clause $F.mgu \vee G.mgu$ .

**Theorem 5.7.4 (J. A. Robinson)**. In the classical predicate logic [$L_1$-$L_{11}$, $L_{12}$-$L_{15}$, MP, Gen], a set of predicate clauses is inconsistent; if and only if Robinson's **Resolution rule** (together with permutation, reduction and **mgu**-rules) allows deriving a contradiction from it.

Why is this (second!) refinement important? Because now, instead of trying out all the possible unifications, we can concentrate on mgu-s that unify two clauses. This allows to further restrict the substitution search space (when compared with Theorem 5.7.2).

The hard part of the proof is inventing of the following

**Lemma 5.7.5.** Any proof $K_1$, $K_2$, ... , $K_s \vdash K$ (all K-s are clauses), where only permutation, reduction, **substitution** and Resolution rules are used, can be converted into a proof $K_1$, $K_2$, ... , $K_s \vdash K'$ such that: a) in the proof, only permutation, reduction, **mgu** and Resolution rules are used; b) K can be obtained from K' by a single (possibly empty) substitution, followed by a chain of permutations and reductions.

**Proof of Theorem 5.7.4.** Assume, the set of clauses $K_1$, $K_2$, ... , $K_s$ is inconsistent. Then, by Theorem 5.7.1, there are two proofs $K_1$, $K_2$, ... , $K_s \vdash B$, $K_1$, $K_2$, ... , $K_s \vdash \neg B$, where where only permutation, reduction, **substitution** and Resolution rules are used. From clauses, these rules allow deriving only of clauses. Hence, B is an atomic formula.

By Lemma 5.7.5, both proofs can be converted into proofs $K_1$, $K_2$, ... , $K_s$ ⊢ $B_1$, $K_1$, $K_2$, ... , $K_s$ ⊢ ¬$B_2$ such that: a) in the proofs, only permutation, reduction, **mgu** and Resolution rules are used; $b_1$) B can be obtained from $B_1$ by a single (possibly empty) substitution (permutations and reductions do not apply to atomic formulas), $b_2$) B can be obtained from $B_2$ by a single (possibly empty) substitution.

Thus, $B_1$ and $B_2$ are unifiable. Let us take their mgu, and apply it. As the result, we obtain a contradiction B', ¬B', where B' is $B_1$.mgu (= $B_2$.mgu). And we have obtained this contradiction from the clauses $K_1$, $K_2$, ... , $K_s$ by using only permutation, reduction, **mgu-** and Resolution rules. Q.E.D.

**Proof Lemma 5.7.5..** Induction by the "height of the resolution tree" (see below).

1. Induction base – no resolutions applied in the proof $K_1$, $K_2$, ... , $K_s$ ⊢ K. Then K is obtained from some $K_i$ by a chain of permutations, reductions and substitutions. Add to this fact an "empty" proof $K_1$, $K_2$, ... , $K_s$ ⊢ $K_i$. And let us compose all the substitutions into a single substitution. Q.E.D.

2. Induction step. Assume, we have the proof $K_1$, $K_2$, ... , $K_s$ ⊢ K, containing at least one resolution. Imagine the last resolution in this proof (C is an atomic formula):

$$\frac{F \vee C , \neg C \vee G}{F \vee G} \; .$$

Then K is obtained from the formula $F \vee G$ by a chain of permutations, reductions and substitutions.

The proofs of the formulas $F \vee C , \neg C \vee G$ possess a "height of the resolution tree" less than the one of the proof $K_1$, $K_2$, ... , $K_s$ ⊢ K. Thus, by induction assumption, we can convert these proofs into permutation-reduction-**mgu-**resolution proofs of some formulas $F_1 \vee C_1 \vee F_2 , G_1 \vee \neg C_2 \vee G_2$ such that:

a) $F \vee C$ can be obtained from $F_1 \vee C_1 \vee F_2$ by a single (possibly empty) substitution $sub_1$, followed by a chain of permutations and reductions. Under $sub_1$, the atomic formula $C_1$ is converted into C.

b) $\neg C \vee G$ can be obtained from $G_1 \vee \neg C_2 \vee G_2$ by a single (possibly empty) substitution $sub_2$, followed by a chain of permutations and reductions. Under $sub_2$, the atomic formula $C_2$ is converted into C.

Since the clauses $F_1 \vee C_1 \vee F_2, G_1 \vee \neg C_2 \vee G_2$ do not contain common variables, the substitutions $sub_1$ and $sub_2$ do not intersect, hence, their union $sub_1 \cup sub_2$ is a substitution sub (a unifier of $C_1$ and $C_2$) such that:

$a_1$) F can be obtained from $(F_1 \vee F_2).sub$ by a chain of permutations and reductions.

$b_1$) G can be obtained from $(G_1 \vee G_2).sub$ by a chain of permutations and reductions.

As we know from the above, the atomic formulas $C_1$ and $C_2$ are unifiable. Let us take their mgu, and apply it to the formulas $F_1 \vee C_1 \vee F_2, G_1 \vee \neg C_2 \vee G_2$. Let us denote by C' the formula $C_1$.mgu (it is equal to $C_2$.mgu). Thus, we have two formulas $F_1$.mgu v C' v $F_2$.mgu and $G_1$.mgu v ¬C' v $G_2$.mgu, and, by permutation and resolution, we can obtain the formula

$$(F_1 \vee F_2).mgu \vee (G_1 \vee G_2).mgu \ .$$

Thus, for the formula $(F_1 \vee F_2).mgu \vee (G_1 \vee G_2).mgu$, we have a permutation-reduction-**mgu-**resolution proof. It remains to show that, from this formula, $F \vee G$ can be obtained by a single substitution, followed by a chain of permutations and reductions.

Since the substitution sub is a unifier of $C_1$ and $C_2$, then, by the definition of mgu, sub=mgu.sub', where sub' is some substitution. Hence,

$a_2$) F can be obtained from $(F_1 \vee F_2).mgu$ by the substitution sub', followed by a chain of permutations and reductions.

$b_2$) G can be obtained from $(G_1 \vee G_2).mgu$ by the substitution sub', followed by a chain of permutations and reductions.

Thus, $F \vee G$ can be obtained from $(F_1 \vee F_2).mgu \vee (G_1 \vee G_2).mgu$ by the substitution sub', followed by a chain of permutations and reductions. Q.E.D.

**Warning!**

Despite its beauty, the resolution method cannot overcome the general complexity problem, mentioned at the end of Section 4.3: by Church-Kalmar Theorem, in the classical predicate logic, the task of reasoning is not algorithmically solvable. And a closer analysis shows that all computer programs implementing resolution method run into loop in many situations, when the formula to be proved is, in fact, unprovable. But in many **practical situations**, experience shows that resolution method solves its task, and – in acceptable time. In particular, Prolog interpreters are using resolution, and are solving many practical tasks in acceptable time!

Further reading:

Logic. Part 2 by **Giorgio Ingargiola.**

**Rajjan Shinghal**. Formal Concepts in Artificial Intelligence. Fundamentals. Chapman&Hall, 1992, 666 pp.

Handbook of Automated Reasoning, ed. by J. A. Robinson and A. Voronkov, Elsevier and MIT Press, 2001, vol. I, II.

About the ubiquity of the above-mentioned unification operation in human and computer reasoning:

**John F. Sowa, Arun K. Majumdar**. Analogical Reasoning. In: *Conceptual Structures for Knowledge Creation and Communication*, Proceedings of ICCS 2003, LNAI 2746, Springer-Verlag, Berlin, 2003, pp. 16-36. (available online).

# 6. Miscellaneous

## 6.1. Negation as Contradiction or Absurdity

The idea behind this approach is as follows: let us define ¬B (i.e. "B is false") as "B implies absurdity". So, let us add to our first order language a predicate constant f (meaning "false", or "absurdity"), and let us replace all negation expressions ¬F by F→f. Then, the three negation axioms will take the following forms:

**L$_9$**: (B→C)→((B→¬C)→¬B),

$$\textbf{L}_9\text{': } (B{\to}C){\to}((B{\to}(C{\to}f)){\to}(B{\to}f)),$$

**L$_{10}$**: ¬B→(B→C),

$$\textbf{L}_{10}\text{': } (B{\to}f){\to}(B{\to}C),$$

**L$_{11}$**: $B \vee \neg B$ ,

$$\textbf{L}_{11}\text{': } B \vee (B \to f) \ .$$

After this, surprisingly, the axiom L$_9$' becomes derivable from L$_1$-L$_2$! Indeed,

   (1)   B→C                            Hypothesis.

   (2)   B→(C→f)                 Hypothesis.

   (3)   B                                 Hypothesis.

   (4)   C→f                          By MP, from (2) and (3)

   (5)   C                                By MP, from (1) and (3)

   (6)   f                                 By MP, from (4) and (5)

Hence, by Deduction Theorem 1,

        [L$_1$, L$_2$, MP] ⊢ (B→C)→((B→(C→f))→(B→f)).

Second observation. The axiom L$_{10}$': (B→f)→(B→C) can be replaced simply by f→C. Indeed, if we assume f→C, then L$_{10}$' becomes derivable:

(1)  B→f                                   Hypothesis.

(2)  B                                     Hypothesis.

(3)  f                                     By MP, from (1) and (2)

(4)  f→C                                   f→C

(5)  C                                     By MP, from (3) and (4)

Hence, by Deduction Theorem 1, $[L_1, L_2, f\to C, MP] \vdash (B\to f)\to(B\to C)$.

Third observation. As we know from Theorem 2.4.9: $[L_1, L_2, L_9, MP] \vdash \neg B\to(B\to\neg C)$, in the minimal logic we can prove 50% of $L_{10}$: "Contradiction implies that all is wrong". After our replacing negations by B→f the formula $(B\to f)\to(B\to(C\to f))$ becomes derivable from $L_1$-$L_2$. Indeed,

(1)  B→f                                   Hypothesis.

(2)  B                                     Hypothesis.

(3)  f                                     By MP, from (1) and (2)

(4)  f→(C→f)                               Axiom $L_1$

(5)  C→f                                   By MP, from (3) and (4)

Hence, by Deduction Theorem 1, $[L_1, L_2, MP] \vdash (B\to f)\to(B\to(C\to f))$.

Thus, we see that $L_1$ (and not $L_9$!) is responsible for the provability of the 50% "crazy" formula $\neg B\to(B\to\neg C)$. Is $L_1$ 50% as "crazy" as $L_{10}$? Yes! Let us compare:

$L_{10}$: $\neg B\to(B\to C)$ states that "Contradiction implies anything".

$L_1$: $B\to(C\to B)$ states that "If B is true, then B follows from anything".

Let us remind our "argument" in favour of $L_{10}$ in <u>Section 1.3</u>: "...we do not need to know, were C "true" or not, if ¬B and B were "true" simultaneously. By assuming that "if ¬B and B were true simultaneously, then anything were true" we greatly simplify our logical apparatus."

Now, similarly: if B is (unconditionally) true, then we do not need to know, follows B from C or not. By assuming that "if B is true, then B follows from anything" we greatly simplify our logical apparatus.

In a sense, the axiom $L_9$ "defines" the negation of the minimal logic, the axioms $L_9$ and $L_{10}$ "define" the negation of the constructive logic, and $L_9$-$L_{11}$ "define" the negation of the classical logic. Is our definition of $\neg B$ as $B \to f$ equivalent to these "definitions"? Yes!

**Theorem 6.1.1.** For any formula F, let us denote by F' the formula obtained from F by replacing all sub-formulas $\neg G$ by $G \to f$. Then, for any formulas $B_1, ..., B_n, C$:

$$[L_1\text{-}L_9, MP]: B_1, ..., B_n \vdash C, \text{ if and only if } [L_1\text{-}L_8, MP]: B'_1, ..., B'_n \vdash C'.$$

**Proof.**

1) $\to$.

Let us consider a proof of $[L_1\text{-}L_9, MP]: B_1, ..., B_n \vdash C$. In this proof:

− let us replace each formula G by its "translation" G',

− before each instance of $L_9$, let us insert a proof of the corresponding instance of $L'_9$ in $[L_1, L_2, MP]$ (see above).

In this way we obtain a proof of $[L_1\text{-}L_8, MP]: B'_1, ..., B'_n \vdash C'$. Indeed,

a) If some formula B is an instance of $L_1$-$L_8$, then B' is an instance of the same axiom (verify!).

b) $(B \to D)'$ is $B' \to D'$, hence, if the initial proof contains a conclusion by MP from B and $B \to D$ to D, then, in the derived proof, it is converted into a conclusion by MP from B' and $B' \to D'$ to D'.

c) If the initial proof contains an instance of $L_9$, then the derived proof contains the corresponding instance of $L'_9$ preceded by its proof in $[L_1, L_2, MP]$.

Q.E.D.

2) $\leftarrow$.

Let us remind the above translation operation: for any formula F, we denoted by F' the formula obtained from F by replacing all sub-formulas $\neg G$ by $G \to f$. Now, let us introduce a kind of a converse operation – the re-translation operation: for any formula F, let us denote by F" the formula obtained from F: a) by replacing all sub-formulas $G \to f$ by $\neg G$, and after this, b) by replacing all the remaining f's (f means "false"!) by $\neg(a \to a)$, where a is some closed formula of the language considered.

Of course, for any formula F, (F')" is F (verify).

**Note**. Replacing f by a formula preceded by negation, is crucial – it will allow applying of Theorem 2.4.9: $[L_1-L_9, MP]$: $\neg B \rightarrow (B \rightarrow \neg C)$ instead of the Axiom $L_{10}$: $\neg B \rightarrow (B \rightarrow C)$.

Now, let us consider a proof of $[L_1-L_8, MP]$: $B'_1, ..., B'_n \vdash C'$. In this proof, let us replace each formula G by its re-translation G". Then C' becomes C, and $B'_1, ..., B'_n$ become $B_1, ..., B_n$, but what about the remaining formulas contained in the proof?

a) Instances of the axioms $L_1-L_8$.

$L_1$: $B \rightarrow (C \rightarrow B)$

If B is not f, then $(B \rightarrow (C \rightarrow B))"$ is $B" \rightarrow (C" \rightarrow B")$, i.e. re-translation yields again an instance of $L_1$.

If B is f, then $(f \rightarrow (C \rightarrow f))"$ is $\neg(a \rightarrow a) \rightarrow \neg C"$. This formula is provable in $[L_1-L_9, MP]$. Indeed,

| (1) | $\neg(a \rightarrow a)$ | Hypothesis. |
|---|---|---|
| (2) | $\vdash \neg(a \rightarrow a) \rightarrow ((a \rightarrow a) \rightarrow \neg C")$ | Theorem 2.4.9, $[L_1-L_9, MP]$. |
| (3) | $\vdash a \rightarrow a$ | Theorem 1.4.1 $[L_1-L_2, MP]$. |
| (4) | $\neg C"$ | By MP, from (1), (2) and (3). |

Thus, re-translation of any instance of $L_1$ is provable in $[L_1-L_9, MP]$.

$L_2$: $(B \rightarrow (C \rightarrow D)) \rightarrow ((B \rightarrow C) \rightarrow (B \rightarrow D))$

If C and D are not f, then re-translation yields again an instance of $L_2$.

If C is f, and D is not, then re-translation yields

$$(B" \rightarrow (\neg(a \rightarrow a) \rightarrow D")) \rightarrow (\neg B" \rightarrow (B" \rightarrow D")).$$

This formula is provable in $[L_1-L_9, MP]$. Indeed,

| (1) | $B" \rightarrow (\neg(a \rightarrow a) \rightarrow D")$ | Hypothesis. |
|---|---|---|
| (2) | $\neg B"$ | Hypothesis. |
| (3) | $B"$ | Hypothesis. |
| (4) | $\neg(a \rightarrow a) \rightarrow D"$ | By MP, from (1) and (3). |

(5)  ⊢¬B"→(B"→¬(a→a))         Theorem 2.4.9 [$L_1$-$L_9$, MP].

(6)  ¬(a→a)                      By MP, from (2), (3) and (5).

(7)  D"                          By MP, from (4) and (6).

Hence, by Deduction Theorem 1,

[$L_1$-$L_9$, MP] ⊢(B"→(¬(a→a)→D"))→(¬B"→(B"→D")).

If D is f, and C is not, then re-translation yields

(B"→¬C")→((B"→C")→¬B").

This formula is provable in [$L_1$-$L_9$, MP]. Indeed,

(1)  B"→¬C"                     Hypothesis.

(2)  B"→C"                      Hypothesis.

(3)  ¬B"                         By MP, from Axiom $L_9$.

Hence, by Deduction Theorem 1,

[$L_1$-$L_9$, MP] ⊢(B"→¬C")→((B"→C")→¬B").

If C and D both are f, then re-translation yields

(B"→¬¬(a→a))→(¬B"→¬B").

This formula is provable in [$L_1$-$L_9$, MP]. Indeed,

(1)  ⊢¬B"→¬B"                   Theorem 1.4.1 [$L_1$-$L_2$, MP].

(2)  ⊢(¬B"→¬B")→(X→(¬B"→¬B"))  Axiom $L_1$, X is B"→¬¬(a→a).

(3)  ⊢X→(¬B"→¬B")              By MP, X is B"→¬¬(a→a).

Thus, re-translation of any instance of $L_2$ is provable in [$L_1$-$L_9$, MP].

**$L_3$:**  $B \wedge C \rightarrow B$

If B is not f, then re-translation yields again an instance of $L_3$.

If B is f, then re-translation yields via $\neg(f \wedge C)$ the formula $\neg(\neg(a \rightarrow a) \wedge C)$. This formula is provable in [$L_1$-$L_9$, MP]. Indeed,

(1)     $\neg(a \to a) \wedge C \to \neg(a \to a)$        Axiom $L_3$.

(2)     $\neg\neg(a \to a) \to \neg(\neg(a \to a) \wedge C)$    From (1), by the Contraposition Law.

(3)   $(a \to a) \to \neg\neg(a \to a)$     Theorem 2.4.4: $[L_1, L_2, L_9, MP] \vdash A \to \neg\neg A$

(4)   $a \to a$     Theorem 1.4.1 $[L_1\text{-}L_2, MP]$.

(5)     $\neg(\neg(a \to a) \wedge C)$        By MP, from (3), (4) and (2).

Thus, re-translation of any instance of $L_3$ is provable in $[L_1\text{-}L_9, MP]$.

**$L_4$:**  $B \wedge C \to C$

Similarly to $L_3$ – re-translation of any instance of $L_4$ is provable in $[L_1\text{-}L_9, MP]$.

**$L_5$:**  $B \to (C \to B \wedge C)$

Re-translation yields again an instance of $L_5$.

**$L_6$:**  $B \to B \vee C$

Re-translation yields again an instance of $L_6$.

**$L_7$:**  $C \to B \vee C$

Re-translation yields again an instance of $L_7$.

**$L_8$:**  $(B \to D) \to ((C \to D) \to (B \vee C \to D))$

If D is not f, then re-translation yields again an instance of $L_8$.

If D is f, then re-translation yields  $\neg B \to (\neg C \to \neg(B \vee C))$ . By Theorem 2.4.10(b), this formula is provable in $[L_1\text{-}L_9, MP]$ .

Thus, re-translation of any instance of $L_8$ is provable in $[L_1\text{-}L_9, MP]$.

Hence, re-translations of all (i.e. $L_1$-$L_8$) axiom instances are provable in $[L_1$-$L_9, MP]$. What about applications of MP in the initial proof? If the initial proof contains a conclusion by MP from B and B→D to D, then the following situations are possible:

a) If B and D are not f, then, in the derived proof, this conclusion is converted into a conclusion by MP from B" and B"→D" to D".

b) If B is f, and D is not, then, in the derived proof, this conclusion is converted into a conclusion by MP from ¬(a→a) and ¬(a→a)→D" to D".

c) If D is f, and B is not, then, in the derived proof, this conclusion is converted into three formulas: B", ¬B", ¬(a→a). To derive ¬(a→a) from B" and ¬B", we can use MP and Theorem 2.4.9:

$$[L_1\text{-}L_9, MP] \vdash \neg B'' \rightarrow (B'' \rightarrow \neg(a \rightarrow a)).$$

d) If B and D are both f, then, in the derived proof, this conclusion is converted into three formulas: ¬(a→a), ¬¬(a→a), ¬(a→a). Simply drop the third formula from the proof.

Thus, the re-translation operation, when applied to all formulas of a proof of $[L_1\text{-}L_8, MP]$: $B'_1, ..., B'_n \vdash C'$, yields a sequence of formulas that are provable in $[L_1\text{-}L_9, MP]$ from hypotheses $B_1, ..., B_n$. Hence, so is C.

Q.E.D.

This completes the proof of Theorem 6.1.1.

**Corollary 6.1.2**. a) A formula C is provable in the minimal propositional logic $[L_1\text{-}L_9, MP]$, if and only if $[L_1\text{-}L_8, MP] \vdash C'$.

b) A formula C is provable in the constructive propositional logic $[L_1\text{-}L_{10}, MP]$, if and only if $[L_1\text{-}L_8, f \rightarrow B, MP] \vdash C'$.

c) A formula C is provable in the classical propositional logic $[L_1\text{-}L_{11}, MP]$, if and only if $[L_1\text{-}L_8, f \rightarrow B, L'_{11}, MP] \vdash C'$.

**Proof.** a) Consider an empty set of hypotheses in Theorem 6.1.1.

b) If $[L_1\text{-}L_{10}, MP] \vdash C$, then $[L_1\text{-}L_9, MP]$: $B_1, ..., B_n \vdash C$, where hypotheses are instances of the axiom $L_{10}$. By Theorem 6.1.1,

$$[L_1\text{-}L_8, MP]: B'_1, ..., B'_n \vdash C'.$$

As established above, $B'_1, ..., B'_n$ can be proved by using the axiom schema f→B, i.e. $[L_1\text{-}L_8, f \rightarrow B, MP] \vdash C'$. Q.E.D.

Now, if $[L_1\text{-}L_8, f \rightarrow B, MP] \vdash C'$, then,

c) If $[L_1\text{-}L_{11}, MP] \vdash C$, then $[L_1\text{-}L_9, MP]$: $B_1, ..., B_n \vdash C$, where hypotheses are instances of the axioms $L_{10}$ and $L_{11}$. Return to case (b). Q.E.D.

**Corollary 6.1.3**. a) A formula C is provable in the minimal predicate logic $[L_1\text{-}L_9, L_{12}\text{-}L_{15}, MP, Gen]$, if and only if $[L_1\text{-}L_8, L_{12}\text{-}L_{15}, MP, Gen] \vdash C'$.

b) A formula C is provable in the constructive predicate logic $[L_1\text{-}L_{10}, L_{12}\text{-}L_{15}, MP, Gen]$, if and only if $[L_1\text{-}L_8, f{\to}B, L_{12}\text{-}L_{15}, MP, Gen] \vdash C'$.

c) A formula C is provable in the classical predicate logic $[L_1\text{-}L_{11}, L_{12}\text{-}L_{15}, MP, Gen]$, if and only if $[L_1\text{-}L_8, f{\to}B, L_{11}', L_{12}\text{-}L_{15}, MP, Gen] \vdash C'$.

**Exercise 6.1.1.** Prove Corollary 6.1.3.