Mihaiela-Ana Lupea Andreea-Diana Mihiş

A Computational Approach to Classical Logics and Circuits

Editura RISOPRINT
Cluj-Napoca • 2015

*Editura **RISOPRINT** este recunoscută de **C.N.C.S.***
(Consiliul Național al Cercetării științifice).
www.risoprint.ro ***www.cncs-uefiscdi.ro***

Descrierea CIP a Bibliotecii Naționale a României
LUPEA, MIHAIELA-ANA
A computational approach to classical logics and circuits / Mihaiela-Ana Lupea, Andreea-Diana Mihiș. - Cluj-Napoca : Risoprint, 2015
Bibliogr.
Index
ISBN 978-973-53-1636-5

I. Mihiș, Andreea-Diana

004.312(075.8)
004.312:62.001.63(075.8)

Director editură: GHEORGHE POP
Consilier editorial: VOICHIȚA-MARIA CLINCI

Referenți științifici:
Prof. univ. dr. DOINA TĂTAR
Prof. univ. dr. GABRIELA CZIBULA

Tiparul executat la:
S.C. ROPRINT® S.R.L.
400 188 Cluj-Napoca • Str. Cernavodă nr. 5-9
Tel./Fax: **0264-590651** • roprint@roprint.ro

PREFACE

The purpose of this book is to present the logical foundations of computer science: classical logics and logic circuits.

Fundamental concepts and results of classical logics are introduced in a formal style and in an explicitly computational way. Applications to automated theorem proving in propositional logic and first-order logic are presented. The studied proof methods are: the resolution method, the semantic tableaux method and the sequent/anti-sequent calculi. The formalization of mathematical reasoning and human reasoning using propositional logic and first-order logic is also an objective of this paper.

The design of logic circuits, based on Boolean functions is an important part of this book.

The paper combines the theoretical presentation of classical logics and logic circuits with numerous examples explained and a rich base of proposed exercises.

Chapter 1 is dedicated to *propositional logic.*The semantic issues discussed are: truth tables, validity, consistency, inconsistency, logical equivalence, logical consequence, normal forms. From a syntactic perspective propositional logic is introduced as an axiomatic (deductive) system, with the purpose of reasoning modeling.

First-order predicate logic is the topic of **chapter 2** of the paper. A Hilbert axiomatic system is used to present predicate calculus in a syntactic approach. The semantics of predicate logic is introduced in order to provide a meaning in terms of the modeled universe for each formula from the language. Normal forms, substitutions and unifiers used in predicative resolution are discussed. The method based on Herbrand's theorem is applied with the aim of reducing the problem of inconsistency of a set of predicate formulas to the problem of inconsistency in propositional logic.

Chapter 3 treats the *semantic tableaux method*, a refutation proof method. The formulas are decomposed in order to determine their models. The classical approach through graphic representation using a binary tree is suggestive, but hard to implement. The new approach, using the TP function allows an easy and efficient implementation of this proof method. An automated theorem prover based on this method, including the corresponding algorithms and implementation details are presented.

Chapter 4 presents two complementary axiomatic systems: *the sequent* and *anti-sequent calculi.* As syntactic and direct proof methods, they are used to check

the validity/derivability and non-validity/non-derivability in propositional logic and first-order logic.

The topic of **chapter 5** is *resolution*, a syntactic and refutation proof method, very efficient and easily to implement. Resolution is introduced as an axiomatic (formal) system and as a procedure. In order to increase the efficiency of the resolution process, the strategies (deletion, set-of-support, unit preference, level saturation, linear) and the refinements (lock, semantic, linear-ordered) of resolution are studied.

In **chapter 6** examples of reasoning modeling in mathematics and daily life, using propositional logic and predicate logic, are presented. An approach of the program verification task using resolution in first-order logic is also described.

Chapter 7 introduces the basic concepts of *Boolean algebras* and *Boolean functions* and presents three *simplification methods* of Boolean functions. Veitch-Karnaugh diagrams graphical method, Quine's analytical method and Moisil's algebraic method are described and applied to numerous examples.

Chapter 8 deals with *logic circuits*. Defined on Boolean algebras, Boolean functions are very useful in the design of logic circuits. Some of the combinational circuits used in the hardware of computers (comparator, adder, subtractor, encoder, decoder) are presented.

By its content, this book is usefull to all those interested in classical logics and logic circuits, fundamental areas of computer science. Professionals in computer science are offered a theoretical basis in the applicative direction of building automated proof systems used in mathematics, software engineering, intelligent agents, robotics, natural language, artificial vision.

We wish to acknowledge our deep appreciation to Prof. Dr. Doina Tătar for many valuable scientific discussions and guidance during all the years of study and research in the field of classical logics. Special thanks for all her constructive comments made during the preparation of this paper.

Cluj-Napoca
2015

Mihaiela-Ana Lupea
Andreea-Diana Mihiş

CONTENTS

INTRODUCTION

The science that studies the principles of reasoning and valid inference is called *logic*. Philosophical logic and mathematical logic are commonly associated with *deductive reasoning*, which determines whether the truth of a conclusion can be determined for an inference rule, based solely on the truth of the set of premises.

In antiquity, earlier philosophers introduced *propositional logic*, but only in the 3rd century B.C. was it developed into a formal logic by the Stoics. In 322 B.C., Aristotle proposed the *syllogism law* that became the dominant model of correct argumentation in philosophy for more than two thousand years. This law was used later as an inference rule in symbolic logic.

The philosopher Gottfried Leibniz (1646 -1716) is considered to be the founder of *symbolic logic*. His research aimed at finding a general decision procedure, but his important results were unknown to the larger logical community.

Completely independent of Leibniz, two well-known mathematicians and logicians reachieved in their works the same results as Leibniz:

- George Boole: *Mathematical Analysis of Logic, giving an algebraic approach to Aristotelian logic* (1847) and *The Laws of Thought* (1854). He applied methods from the field of symbolic algebra to logic.
- Augustus De Morgan: *Formal Logic* (1847). His major contribution was to the development of the theory of relations and the rise of modern symbolic, or mathematical, logic.

In 1879 Gottlob Frege proposed a formal system called *predicate* or *first-order logic* (FOL), introducing quantified statements and proposing the notion of a *proof* in terms that are still accepted today.

Propositional logic and predicate logic are called *classical logics* and they model a valid type of deductive reasoning. The decision problems in these logics are: to check the validity of a statement and to check if a *conclusion* is derivable (inferable) from a set of statements (*axioms* and *hypotheses*). The inferential process is a monotonic one, meaning that once a conclusion is deduced (derived) from a set of hypotheses (premises), new premises will not invalidate it.

Once the formal framework (syntax and semantics) of logic was developed, the efforts of researchers focused on finding efficient *proof methods* that would solve the decision problems and implement them. The proof methods are classified into *direct methods* and *refutation methods*. The direct methods build the proof of the conclusion directly from the axioms and the hypotheses using the inference rules. The refutation methods are based on the 'reductio ad absurdum' principle and the idea is to show that the negation of the conclusion together with the hypotheses and the axioms lead to a contradiction.

Introduction

The invention of *truth-table*, an important tool in propositional logic, is of controversial attribution because many mathematicians-logicians (Frege, Russell, Philo, Boole, Peirce, Schröder, Łukasiewicz, Whitehead, Venn, Lewis) had ideas related to truth-tables. However, the well-known tabular structure is credited to either Ludwig Wittgenstein (1922) or Emil Post (1921).

In 1936 G.K.Gentzen introduced *natural deduction*, a direct and syntactic method, based on an axiomatic system that formalizes the deductive (inferential) process. As an improvement of this method, two complementary axiomatic systems *sequent* and *anti-sequent calculi* were developed and employed to prove the validity/derivability and non-validity/non-derivability respectively in classical logics.

In 1965 J.A.Robinson proposed a refutation and syntactic proof method, simple and easy to implement, called *resolution*. Later, this method was refined (semantic resolution, linear resolution, lock resolution) in order to increase the efficiency.

The *semantic tableaux method* introduced by R. Smullyan in 1968 is based on semantic considerations. The proof by contradiction of the conclusion consists of decomposing the conjunction of the hypotheses and the negation of the conclusion, searching for models and showing that they do not exist.

Automated Theorem Proving (ATP) deals with the development of computer programs which show that some statement (the *conjecture*) is a *logical consequence* of a set of statements (the *axioms* and the *hypotheses*). These automated systems were used in a lot of domains such as mathematics (*EQP, Otter, Geometry Expert*), software generation (*KIDS, AMPHION*), software verification (*KIV, PVS*), hardware verification (*ACL2, HOL, ANALYTICA)*. As potential fields we enumerate biology, social science, medicine, commerce.
Also developed were dedicated (educational) automated theorem provers:

- based on semantic tableaux method: 3TAP, pTAP, leanTAP, Cassandra;
- based on resolution method: OTTER, PCPROVE, AMPHION, Jape;
- based on semantic trees + Herbrand theorem: HERBY;
- based on model elimination calulus: SETHEO;

In 1938 Claude Shannon proved that a two-valued binary Boolean algebra can describe the operations of two-valued electrical switching circuits. Propositional logic is used to minimize the number of gates in a circuit, and to show the equivalence of *combinational circuits*.

The Romanian mathematician Grigore Moisil invented the three-stable circuits and had important contributions in the fields of algebraic logic and differential equations. Moisil used propositional logic to minimize Boolean functions.
In modern times *Boolean algebras* and *Boolean functions* are indispensable in the design of computer chips and digital circuits.

A computational approach, providing methods and algorithms to solve different tasks in classical logics and circuits is the specific purpose of this book.

1. PROPOSITIONAL LOGIC

In this chapter the propositional logic is presented, first from a semantic perspective and then as an axiomatic system. This formalism is useful in reasoning modeling and the design of logic circuits. We used as references the following papers [1, 2, 9, 16, 19, 23, 37, 38, 40, 42, 43, 47, 48, 59, 60, 61, 62, 63, 64].

1.1. Syntax

The *syntax* introduces the entities used to define well-formed propositional formulas.

- $\Sigma_P = Var\_propos \cup \{F, T\} \cup Connectives \cup \{(,)\}$ is the ***vocabulary***;
- $Var\_propos = \{p, q, r, ...\}$ is a finite ***set of propositional variables***;
- $Connectives = \{\neg$ (negation), $\wedge$ (conjunction), $\vee$ (*disjunction*), $\rightarrow$ (*implication*), $\leftrightarrow$ (*equivalence*)$\}$:
  The negation is a unary connective and all the others are binary connectives.
  The decreasing order of precedence of the connectives is as follows:
  $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$.
- F_P is the ***set of well-formed formulas*** built using the propositional variables, the connectives and the parentheses (to avoid ambiguity).
  example: $(p \rightarrow \neg q) \wedge (r \vee q \leftrightarrow p) \wedge s$ is a propositional formula.

1.2. Semantics of propositional logic

Logical propositions are models of propositional assertions from natural language, which can be "true" or "false".

The aim of the semantics is to give a meaning (to assign a truth value) to the propositional formulas. The ***semantic domain*** is the set of ***truth values***: { F (false), T (true)}, which satisfy the relations: $\neg F = T, \neg T = F$.

New connectives $\uparrow$ ("*nand*"), $\downarrow$ ("*nor*"), $\oplus$ ("*xor*") are introduced using the following definitions:

$p \uparrow q := \neg(p \wedge q)$,

$p \downarrow q := \neg(p \vee q)$,

$p \oplus q := \neg(p \leftrightarrow q)$

These new connectives are used in the design of logic circuits.

The semantics of the connectives are provided by the following ***truth tables***:

p	q	$\neg p$	$p \wedge q$	$p \vee q$	$p \rightarrow q$	$p \leftrightarrow q$	$p \uparrow q$	$p \downarrow q$	$p \oplus q$
T	T	F	T	T	T	T	F	F	F
T	F	F	F	T	F	F	T	F	T
F	T	T	F	T	T	F	T	F	T
F	F	T	F	F	T	T	T	T	F

Remarks:

- A conjunction is *true* exactly when both its operands are true. As a generalization, the conjunction $p_1 \wedge p_2 \wedge \ldots \wedge p_n$ is *true* exactly when all its n operands are *true*.
- A disjunction ("*inclusive or*") is *false* only when both its operands are *false*. As a generalization, the conjunction $p_1 \vee p_2 \vee \ldots \vee p_n$ is *false* only when all its n operands are *false*.
- The implication $p \rightarrow q$ is *false* only when the hypothesis p is true and the conclusion q is *false* (*true* cannot imply *false*).
- The equivalence $p \leftrightarrow q$ is true only when p and q have the same truth value.
- The connective $\oplus$ ("*exclusive or*") is the negation of equivalence and it is true only when one operand is *true* and the other one is *false*.

Definition 1.1.

An ***interpretation*** of a formula $U(p_1, p_2, \ldots, p_n) \in F_P$ is a function $i: \{p_1, p_2, \ldots, p_n\} \rightarrow \{F, T\}$ which can be extended to $i: F_P \rightarrow \{F, T\}$ using the following relations:

$i(\neg p) = \neg i(p)$

$i(p \wedge q) = i(p) \wedge i(q)$

$i(p \vee q) = i(p) \vee i(q)$

$i(p \rightarrow q) = i(p) \rightarrow i(q)$

$i(p \leftrightarrow q) = i(p) \leftrightarrow i(q)$

Interpretations assign truth values to propositional variables and using the semantics of the connectives evaluate formulas assigning truth values to them. The semantics is compositional, meaning that the truth value of a formula is obtained from the truth values of its subformulas.

Definition 1.2. (semantic concepts)

Let $U(p_1, p_2, \ldots, p_n)$ be a propositional formula.

1. An interpretation i which evaluates the formula U as *true*, $i: \{p_1, \ldots, p_n\} \rightarrow \{T, F\}$ such that $i(U) = T$, is called a ***model*** of U.

2. An interpretation i which evaluates the formula U as *false*, $i:\{p_1,...,p_n\}\to\{T,F\}$ such that $i(U)=F$, is called an ***anti-model*** of U.
3. A *formula* U is called ***consistent*** (***satisfiable***) if it has at least one model: $\exists i:\{p_1,...,p_n\}\to\{T,F\}$ such that $i(U)=T$.
4. The *formula* U is called ***valid*** (***tautology***) and we use the notation: $\models U$, if U is evaluated as true in all its interpretations: $\forall i:\{p_1,...,p_n\}\to\{T,F\}, i(U)=T$. All 2^n interpretations of U are ***models*** of U.
5. The *formula* U is called ***inconsistent*** (***unsatisfiable***) if U does not have any model, thus U is evaluated as false in all its interpretations: $\forall i:\{p_1,...,p_n\}\to\{T,F\}, i(U)=F$. All 2^n interpretations of U are ***anti-models*** of U.
6. The *formula* U is called ***contingent*** if U is consistent, but it is not valid. A contingent formula has at least one model and at least one anti-model.

Remarks:

- In order to evaluate a propositional formula $U(p_1,p_2,...,p_n)$ in all its 2^n interpretations, a truth table is built. The table has 2^n lines (rows), corresponding to all interpretations and the first n columns are filled with all the possible assignments of *true* and *false* values to the n propositional variables. The column of U is obtained using the semantics and the precedence of the connectives.
- If the truth table (column) of U contains only "T", then U is a *tautology*.
- If the truth table (column) of U contains only "F", then U is an *inconsistent formula*.
- For a propositional formula, its *models* are the interpretations (the table's rows) which evaluate the formula as *true* and its *anti-models* are the interpretations (the table's rows) which evaluate the formula as *false*.

The notion of *logical consequence* captures the essence of logical thinking and it is a generalization of the *tautology* notion, as we shall see in the following definition.

Definition 1.3.
The formula $V\in F_P$ is a ***logical consequence*** of the formula $U\in F_P$, notation: $U\models V$, if all the models of U are also models of V:

$\forall i:F_P\to\{T,F\}$ such that $i(U)=T$, we have that $i(V)=T$.

Definition 1.4.
The formulas $U\in F_P$ and $V\in F_P$ are ***logically equivalent***, notation: $U\equiv V$, if U and V have identical truth tables: $\forall i:F_P\to\{T,F\}$ we have that $i(U)=i(V)$.

Note that "$\models$" and "$\equiv$" are meta-symbols used to express logical relations between propositional formulas.

Example 1.1.
Build the truth tables of the following formulas:

$U(p,q,r) = (\neg p \vee q) \wedge (r \vee p)$

$V(p,q,r) = (\neg p \wedge r) \vee (q \wedge r) \vee (q \wedge p)$

$W(p,q,r) = (p \uparrow (\neg p \wedge q)) \vee r$

$Z(p,q,r) = p \wedge ((\neg q \vee r) \downarrow q)$

In order to evaluate U, the formula is decomposed in two subformulas: $\neg p \vee q$ and $r \vee p$, which are evaluated and then we apply the conjunction between their corresponding truth tables (columns). In the same manner the columns of V,W,Z are built.

	p	q	r	$\neg p \vee q$	$r \vee p$	$U(p,q,r)$	$V(p,q,r)$	$W(p,q,r)$	$Z(p,q,r)$
i_1	T	T	T	T	T	T	T	T	F
i_2	T	T	F	T	T	T	T	T	F
i_3	T	F	T	F	T	F	F	T	F
i_4	T	F	F	F	T	F	F	T	F
i_5	F	T	T	T	T	T	T	T	F
i_6	F	T	F	T	F	F	F	T	F
i_7	F	F	T	T	T	T	T	T	F
i_8	F	F	F	T	F	F	F	T	F

- U,V,W and Z have $2^3 = 8$ interpretations: $i_1, i_2, ..., i_8$.
- The formula $U(p,q,r)$ is contingent, having four models: i_1, i_2, i_5, i_7 and four anti-models: i_3, i_4, i_6, i_8.
  $i_1 : \{p,q,r\} \to \{T,F\}$, $i_1(p) = T$, $i_1(q) = T$, $i_1(r) = T$ and $i_1(U) = T$
  $i_6 : \{p,q,r\} \to \{T,F\}$, $i_6(p) = F$, $i_6(q) = T$, $i_6(r) = F$ and $i_6(U) = F$
- The formula $W(p,q,r)$ is a tautology, all its eight interpretations are models.
- The formula $Z(p,q,r)$ is inconsistent being evaluated as *false* in all its eight interpretations, which are the anti-models of Z.
- $U \equiv V$, because U and V have identical truth tables (columns).
- $U \models \neg p \vee q$, because all the models (i_1, i_2, i_5, i_7) of U are also models of the formula: $\neg p \vee q$.

1.3. Logical equivalences

The following logical equivalences express properties of connectives, relations between them and they are used to transform syntactically a propositional formula, preserving its meaning (semantics).

- ***Simplification laws:***

$\neg\neg U \equiv U$	and	$U \to U \equiv T$
$U \wedge \neg U \equiv F$	and	$U \vee \neg U \equiv T$
$T \wedge U \equiv U$	and	$F \vee U \equiv U$
$U \to T \equiv T$	and	$U \to F \equiv \neg U$
$T \to U \equiv U$	and	$F \to U \equiv T$
$U \leftrightarrow T \equiv U$	and	$U \leftrightarrow F \equiv \neg U$
$U \oplus T \equiv \neg U$	and	$U \oplus F \equiv U$
$U \leftrightarrow U \equiv T$	and	$U \oplus U \equiv F$

- ***Idempotency laws:***

$U \wedge U \equiv U$	and	$U \vee U \equiv U$

- ***Commutative laws:***

$U \wedge V \equiv V \wedge U$	and	$U \vee V \equiv V \vee U$
$U \uparrow V \equiv V \uparrow U$	and	$U \downarrow V \equiv V \downarrow U$
$U \leftrightarrow V \equiv V \leftrightarrow U$	and	$U \oplus V \equiv V \oplus U$

- ***Absorption laws:***

$U \wedge (U \vee V) \equiv U$	and	$U \vee (U \wedge V) \equiv U$

- ***Associative laws:***

$(U \wedge V) \wedge Z \equiv U \wedge (V \wedge Z)$	and	$(U \vee V) \vee Z \equiv U \vee (V \vee Z)$

- ***Distributive laws:***

$U \wedge (V \vee Z) \equiv (U \wedge V) \vee (U \wedge Z)$	and	$U \vee (V \wedge Z) \equiv (U \vee V) \wedge (U \vee Z)$

- ***De Morgan's laws:***

$\neg(U \wedge V) \equiv \neg U \vee \neg V$	and	$\neg(U \vee V) \equiv \neg U \wedge \neg V$

- ***Relations between connectives:***

$U \to V \equiv \neg U \vee V$	and	$U \to V \equiv \neg(U \wedge \neg V)$
$U \to V \equiv U \leftrightarrow (U \wedge V)$	and	$U \to V \equiv V \leftrightarrow (U \vee V)$
$U \leftrightarrow V \equiv (U \to V) \wedge (V \to U)$	and	$U \oplus V \equiv \neg(U \to V) \vee \neg (V \to U)$
$U \leftrightarrow V \equiv (U \vee V) \to (U \wedge V)$		
$U \vee V \equiv \neg(\neg U \wedge \neg V)$	and	$U \wedge V \equiv \neg(\neg U \vee \neg V)$
$U \vee V \equiv \neg U \to V$	and	$U \wedge V \equiv \neg(U \to \neg V)$
$\neg U \equiv U \uparrow U$	and	$\neg U \equiv U \downarrow U$
$U \vee V \equiv (U \uparrow U) \uparrow (V \uparrow V)$	and	$U \wedge V \equiv (U \downarrow U) \downarrow (V \downarrow V)$
$U \vee V \equiv (U \downarrow V) \downarrow (U \downarrow V)$	and	$U \wedge V \equiv (U \uparrow V) \uparrow (U \uparrow V)$

The duality principle: For every logical equivalence $U \equiv V$ containing only the connectives $\neg, \wedge, \vee, \uparrow, \downarrow, \leftrightarrow, \oplus$, there is another logical equivalence $U' \equiv V'$, where U', V' are formulas obtained from U, V by interchanging the truth values (T, F) and the connectives from the following pairs: $(\wedge, \vee)$, $(\uparrow, \downarrow)$, $(\leftrightarrow, \oplus)$.

Note that some of the above laws are pairs of ***dual logical equivalences***.
Dual connectives: $(\wedge, \vee)$, $(\uparrow, \downarrow)$, $(\leftrightarrow, \oplus)$.
Dual truth values: (T, F).

Definition 1.5.
A set of connectives is *functionally complete* if there is no truth table that cannot be expressed as a formula involving only these connectives. All the other connectives can be expressed using the connectives from such a set.

The following sets of propositional connectives are functionally complete.

1. $\{\neg, \wedge\}$;
2. $\{\neg, \vee\}$;
3. $\{\neg, \rightarrow\}$;
4. $\{\oplus, \wedge\}$;
5. $\{\oplus, \vee\}$;
6. $\{\oplus, \rightarrow\}$;
7. $\{\uparrow\}$;
8. $\{\downarrow\}$;

The following definitions show that from the semantic point of view, a set of formulas is equivalent to the conjunction of its elements.

Definition 1.6.
Let $U_1, U_2, ..., U_n, V$ be propositional formulas.

1. The set $\{U_1, U_2, ..., U_n\}$ is called ***consistent*** if the formula $U_1 \wedge U_2 \wedge ... \wedge U_n$ is consistent:
   $\exists i : F_P \rightarrow \{T, F\}$ such that $i(U_1 \wedge U_2 \wedge ... \wedge U_n) = T$.
2. The set $\{U_1, U_2, ..., U_n\}$ is called ***inconsistent*** if the formula $U_1 \wedge U_2 \wedge ... \wedge U_n$ is inconsistent:
   $\forall i : F_P \rightarrow \{T, F\}$, $i(U_1 \wedge U_2 \wedge ... \wedge U_n) = F$.
3. The formula V, called ***conclusion***, is a ***logical consequence*** of the set $\{U_1, U_2, ..., U_n\}$ of formulas called ***premises (hypotheses, facts)***, notation: $U_1, U_2, ..., U_n \models V$, if $\forall i : F_P \rightarrow \{T, F\}$ such that $i(U_1 \wedge U_2 \wedge ... \wedge U_n) = T$, we have $i(V) = T$.
   The models of the set of premises are also models of the conclusion.

Theorem 1.1.
Let $U_1, U_2, ..., U_n, U, V$ be propositional formulas.

1. $\models U$ *if and only if* $\neg U$ is inconsistent.
   A formula is a tautology *if and only if* its negation is inconsistent.
2. $U \models V$ *if and only if* $\models U \rightarrow V$ *if and only if* the set $\{U, \neg V\}$ is inconsistent.

3. $U \equiv V$ *if and only if* $\models U \leftrightarrow V$.
   U and V are logically equivalent *if and only if* the formula
   $U \leftrightarrow V$ is a tautology.
4. $U_1, U_2, ..., U_n \models V$ *if and only if*
   $\models U_1 \wedge U_2 \wedge ... \wedge U_n \rightarrow V$ *if and only if*
   the set $\{U_1, U_2, ..., U_n, \neg V\}$ is inconsistent.

The proofs of the previous assertions are simple and we shall prove only some of them.

Proofs:

2. **If** $U \models V$ **then** $\models U \rightarrow V$.
   If $U \models V$, using the definition of logical consequence relation (Definition 1.3), then: $\forall i : F_P \rightarrow \{T, F\}$ such that $i(U) = T$ we have that $i(V) = T$.
   We have to evaluate the formula $U \rightarrow V$ in all the possible interpretations.
   There are two cases: a) and b).
   a) $i(U) = T$ and using the hypothesis $i(V) = T$,
      we have that $i(U \rightarrow V) = i(U) \rightarrow i(V) = T \rightarrow T = T$
   b) $i(U) = F$ thus $i(U \rightarrow V) = i(U) \rightarrow i(V) = F \rightarrow i(V) = T$
   From a) and b) we have that $\forall i : F_P \rightarrow \{T, F\}, i(U \rightarrow V) = T$, therefore $\models U \rightarrow V$.
3. **If** $U \equiv V$ **then** $\models U \leftrightarrow V$.
   If $U \equiv V$, using the definition of logical equivalence relation (Definition 1.4):
   $\forall i : F_P \rightarrow \{T, F\}$, $i(U) = i(V)$, then
   $\forall i : F_P \rightarrow \{T, F\}$, $i(U \leftrightarrow V) = T$ and according to the definition of a tautology we have that $\models U \leftrightarrow V$
4. **If** the set $\{U_1, U_2, ..., U_n, \neg V\}$ is inconsistent **then** $U_1, U_2, ..., U_n \models V$ holds.
   If the set $\{U_1, U_2, ..., U_n, \neg V\}$ is inconsistent, using the definition of inconsistency of a set of formulas:
   $\forall i : F_P \rightarrow \{T, F\}$, $i(U_1 \wedge U_2 \wedge ... \wedge U_n \wedge \neg V) = F$, then
   $\forall i : F_P \rightarrow \{T, F\}$ there are two possible cases: a) and b).
   a) $\exists k, 1 \le k \le n$, such that $i(U_k) = F$ then
      $i(U_1 \wedge U_2 \wedge ... \wedge U_n) = F$, thus i is an anti-model of the set of premises of the logical consequence: $U_1, U_2, ..., U_n \models V$ and it is an irrelevant case.
   b) $\forall k, 1 \le k \le n$, $i(U_k) = T$ and $i(\neg V) = F$ then
      $i(U_1 \wedge U_2 \wedge ... \wedge U_n) = T$ and $i(V) = T$ then

any model i of the set of premises $\{U_1,U_2,...,U_n\}$ is also a model of the conclusion V, so according to the definition of logical consequence relation we have that: $U_1,U_2,...,U_n \models V$ holds.

Remark:
The assertions 1, 2 and 4 from Theorem 1.1 are used in the refutation proof methods, such as *resolution* and *semantic tableaux method* and they model the proof by contradiction ("reductio ad absurdum").

Theorem 1.2.
Let $S=\{U_1,U_2,...,U_n\}$ be a set of propositional formulas.

1. If S is a consistent set, then $\forall j, 1\le j\le n,\ S-\{U_j\}$ is a consistent set.
2. If S is a consistent set and V is a valid formula, then $S\cup\{V\}$ is a consistent set.
3. If S is an inconsistent set and U_j is valid, where $1\le j\le n$, then the set $S-\{U_j\}$ is inconsistent.
4. If S is an inconsistent set, then $\forall V\in F_P$, the set $S\cup\{V\}$ is inconsistent.

Proofs: The proofs of the above assertions are simple, the Definition 1.6 is used.

1. If S is a consistent set then
   $\exists i: F_P\to\{T,F\}$ such that $i(U_1\wedge U_2\wedge...\wedge U_n)=T$ then
   $\exists i: F_P\to\{T,F\}$ such that $\forall k, 1\le k\le n,\ i(U_k)=T$ then
   $\exists i: F_P\to\{T,F\}$ such that $\forall j, 1\le j\le n,$
   $$i(U_1\wedge U_2\wedge...\wedge U_{j-1}\wedge U_{j+1}\wedge...\wedge U_n)=T,$$
   therefore $\forall j, 1\le j\le n,\ S-\{U_j\}$ is a consistent set.
4. If S is an inconsistent set then
   $\forall i: F_P\to\{T,F\}$, $i(U_1\wedge U_2\wedge...\wedge U_n)=F$ then
   $\forall i: F_P\to\{T,F\}$ and $\forall V\in F_P$, $i(U_1\wedge U_2\wedge...\wedge U_n\wedge V)=F$,
   therefore the set $S\cup\{V\}$ is inconsistent.

Theorem 1.3.
Let R,S be sets of propositional formulas and U,V,Z, be propositional formulas.
The logical consequence relation has the following properties:

1. ***monotonicity***:
   if $R\models U$ and $R\subseteq S$ then $S\models U$;
2. ***cut***:
   if $S\models V_j$, $\forall j\in\{1,...,n\}, n\in\mathbf{N}$ and $S\cup\{V_1,V_2,...,V_n\}\models U$ then $S\models U$;

3. ***transitivity***:
   if $S \models U$ and $\{U\} \models V$ then $S \models V$;
4. ***conjunction in conclusions (right "and")***:
   if $S \models U$ and $S \models V$ then $S \models U \wedge V$;
5. ***disjunction in premises (left "or")***:
   if $S \cup \{U\} \models Z$ and $S \cup \{V\} \models Z$ then $S \cup \{U \vee V\} \models Z$;

Proofs: These properties can be easily proved using the definition of logical consequence relation.

1. ***monotonicity***:
   If $R \models U$, then $\forall i : F_P \to \{T,F\}$ such that $i(R) = T$, we have that $i(U) = T$.
   If $R \subseteq S$ then $\forall i : F_P \to \{T,F\}$ such that $i(S) = T$ then $i(R) = T$.
   From these two relationships, we have that $\forall i : F_P \to \{T,F\}$ such that $i(S) = T$ then $i(R) = T$, so $i(U) = T$ and thus $S \models U$.
2. ***cut***:
   If $S \models V_j$, $\forall j \in \{1,...,n\}, n \in \mathbf{N}$, then $\forall i : F_P \to \{T,F\}$ such that $i(S) = T$, we have that $i(V_j) = T$.
   So, $\forall i : F_P \to \{T,F\}$ such that $i(S) = T$, $\forall j \in \{1,...,n\}, n \in \mathbf{N}$, we have that $i(V_j) = T$.
   In other words, $\forall i : F_P \to \{T,F\}$ such that $i(S) = T$, $i(S \cup \{V_1, V_2, ..., V_n\}) = T$.
   But $S \cup \{V_1, V_2, ..., V_n\} \models U$, which means that $\forall i : F_P \to \{T,F\}$ such that $i(S \cup \{V_1, V_2, ..., V_n\}) = T$, we have that $i(U) = T$.
   From the last two statements we obtain that $\forall i : F_P \to \{T,F\}$ such that $i(S) = T$, $\forall j \in \{1,...,n\}, n \in \mathbf{N}$, we have: $i(S \cup \{V_1, V_2, ..., V_n\}) = T$ and that $i(U) = T$.
   So, $\forall i : F_P \to \{T,F\}$ such that $i(S) = T$, we have that $i(U) = T$, which is $S \models U$.
3. ***transitivity***:
   If $S \models U$, then $\forall i : F_P \to \{T,F\}$ such that $i(S) = T$, we have that $i(U) = T$.
   If $\{U\} \models V$, then $\forall i : F_P \to \{T,F\}$ such that $i(U) = T$, we have that $i(V) = T$.
   So, $\forall i : F_P \to \{T,F\}$ such that $i(S) = T$, we have that $i(U) = T$ and also $i(V) = T$, which is $S \models V$.

1.4. Normal forms in propositional logic

Some of the proof methods need as input data formulas having a certain character of "normal" or "canonical" form.

Definition 1.7.

1. A ***literal*** is a propositional variable or its negation. ($p, \neg q, r$).
2. A ***clause*** is a disjunction of a finite number of literals. (examples: p, $\neg p \vee q$, $r \vee q \vee s$).
3. A ***cube*** is a conjunction of a finite number of literals. (examples: q, $p \wedge \neg q$, $r \wedge s \wedge p$).
4. The ***empty clause***, denoted by $\square$, is the clause without any literal and it is the only inconsistent clause.
5. A formula is in ***disjunctive normal form*** (DNF), if it is written as a disjunction of cubes: $\vee_{i=1}^{p} (\wedge_{j=1}^{q} l_{ij})$, where l_{ij} are literals.
6. A formula is in ***conjunctive normal form*** (CNF), if it is written as a conjunction of clauses: $\wedge_{i=1}^{n} (\vee_{j=1}^{m} l_{ij})$, where l_{ij} are literals.

Example 1.2.

p - DNF with one unit cube;

$p \vee \neg q \vee r$ - DNF with three unit cubes;

$p \wedge q$ - DNF with one cube;

$p \vee (q \wedge r) \vee (\neg p \wedge \neg r \wedge s)$ - DNF with three cubes.

Example 1.3.

p - CNF with one unit clause;

$p \vee \neg q \vee r$ - CNF with one clause;

$p \wedge q$ - CNF with two unit clauses;

$\neg p \wedge (q \vee \neg r) \wedge (p \vee r \vee \neg s)$ - CNF with three clauses.

Property:

Let $\{l_1, l_2, ..., l_n\}$ be a set of literals. The following assertions are equivalent:

1. The clause $\vee_{j=1}^{n} l_j$ is a tautology.
2. The cube $\wedge_{j=1}^{n} l_j$ is inconsistent.
3. The set $\{l_1, l_2, ..., l_n\}$ of literals contains at least one pair of opposite literals: $\exists j, k \in \{1, .., n\}$ such that $l_j = \neg l_k$.

Example 1.4.

1. The *clause* $U = p \vee q \vee r \vee \neg p$ is a tautology ($U \equiv T$) since $p, \neg p$ are opposite literals.
2. The *cube* $V = p \wedge q \wedge r \wedge \neg p$ is an inconsistent formula ($U \equiv F$), because it is a conjunction with $p, \neg p$ as opposite literals.

Theorem 1.4.
Every propositional formula admits an equivalent conjunctive normal form (CNF) and an equivalent disjunctive normal form (DNF).

The ***normalization algorithm*** consists of transformations which preserve the logical equivalence and are applied to the initial formula in order to obtain the corresponding normal form.

Step 1:

- The formulas of "$U \to V$" type are replaced by the logical equivalent form $\neg U \vee V$.
- The formulas of "$U \leftrightarrow V$" type are replaced by the logical equivalent form $(\neg U \vee V) \wedge (\neg V \vee U)$.

Step 2:

- De Morgan's laws are applied: the negations are pushed in until they apply only to propositional variables.
- Multiple negations are eliminated by the reduction rule: $\neg\neg U \equiv U$.

Step 3:

- The distributive laws are applied.

Theorem 1.5.

1. A formula in CNF is a tautology if and only if all its clauses are tautologies.
2. A formula in DNF is inconsistent if and only if all its cubes are inconsistent.

Remarks:

- The first part of the above theorem provides a direct method to prove that a formula is a tautology.
- The DNF of a propositional formula provides all the models of that formula, finding all the interpretations which evaluate, one by one, the cubes as true.
- The CNF of a propositional formula provides all the anti-models of that formula, finding all the interpretations which evaluate, one by one, the clauses as false.

Dual concepts: clause-cube, DNF-CNF.

Example 1.5.
Write the equivalent CNF of the second axiom of propositional logic:
$A_2 = ((U \to (V \to Z)) \to ((U \to V) \to (U \to Z))$.

The normalization algorithm is applied as follows:

$A_2 = ((U \overset{1}{\to} (V \overset{2}{\to} Z)) \overset{3}{\to} ((U \overset{4}{\to} V) \overset{5}{\to} (U \overset{6}{\to} Z)) \equiv$

(replace $\to$ from the inside formulas, denoted by 2,4 and 6)

$\equiv ((U \to (\neg V \vee Z)) \to ((\neg U \vee V) \to (\neg U \vee Z)) \equiv$

(replace the main connective $\to$, denoted by 3)

$\equiv \neg(U \to (\neg V \vee Z)) \vee ((\neg U \vee V) \to (\neg U \vee Z)) \equiv$

(replace both $\to$ connectives, denoted by 1 and 5)

$\equiv \neg(\neg U \vee \neg V \vee Z) \vee (\neg(\neg U \vee V) \vee \neg U \vee Z) \equiv$

(apply De Morgan's laws)

$\equiv (U \wedge V \wedge \neg Z) \vee (U \wedge \neg V) \vee \neg U \vee Z$ – DNF with 4 cubes

(apply the distributive laws)

$\equiv (U \vee \underline{U} \vee \underline{\neg U} \vee Z) \wedge (\underline{U} \vee \neg V \vee \underline{\neg U} \vee Z) \wedge (V \vee \underline{U} \vee \underline{\neg U} \vee Z) \wedge$

$\wedge (\underline{V} \vee \underline{\neg V} \vee \neg U \vee Z) \wedge (\underline{\neg Z} \vee U \vee \neg U \vee \underline{Z}) \wedge (\underline{\neg Z} \vee \neg V \vee \neg U \vee \underline{Z}) \equiv T$

We have obtained a CNF with 6 clauses which are tautologies. Each clause contains at least one pair of opposite formulas (underlined).
Thus, according to the previous theorem, the formula A_2 is a tautology.

Example 1.6.
Write the equivalent DNF of the formula: $X = \neg A_1 = \neg(U \to (V \to U))$

$X = \neg(U \to (V \to U)) \equiv \neg(\neg U \vee (\neg V \vee U)) \equiv U \wedge \neg(\neg V \vee U)) \equiv \underline{U} \wedge V \wedge \underline{\neg U}$

We have obtained a CNF (with three unit clauses) or a DNF (with one cube). DNF contains an inconsistent cube, having a pair of opposite formulas, thus X is an inconsistent formula and A_1 (the first axiom of the propositional logic) is a tautology.

Example 1.7.
Using the appropriate normal form write all the models of the propositional formula: $U = (p \wedge q \to r) \to (p \to r) \wedge q$.
We apply the normalization algorithm:

$U = (p \wedge q \to r) \to (p \to r) \wedge q \equiv \neg(p \wedge q \to r) \vee (p \to r) \wedge q \equiv$

$\equiv \neg(\neg(p \wedge q) \vee r) \vee (\neg p \vee r) \wedge q \equiv (p \wedge q \wedge \neg r) \vee (\neg p \vee r) \wedge q \equiv$

$\equiv (p \wedge q \wedge \neg r) \vee (\neg p \wedge q) \vee (r \wedge q)$ – DNF with 3 cubes

The models of U are the interpretations which evaluate one by one the cubes of DNF as true. In a cube we assign the truth value T to all its literals.

Cube: $p \wedge q \wedge \neg r$ provides one model assigning to all the literals $\{ p, q, \neg r \}$ of the cube the truth value T.

$i_1 : \{p,q,r\} \to \{T,F\}, i_1(p) = T, i_1(q) = T, i_1(r) = F$ and $i_1(p \wedge q \wedge \neg r) = T$

Cube: $\neg p \wedge q$.

The propositional variable r is missing in the cube's expression, so the truth value of the cube does not depend on the truth value assigned to r: we need to assign to p the value F, to q the value T, but to r we can assign either T or F such that the cube is evaluated as true.

Therefore this cube provides the following models:

$i_2 : \{p,q,r\} \to \{T,F\}, i_2(p) = F, i_2(q) = T, i_2(r) = T$ and $i_2(\neg p \wedge q) = T$

$i_3 : \{p,q,r\} \to \{T,F\}, i_3(p) = F, i_3(q) = T, i_3(r) = F$ and $i_3(\neg p \wedge q) = T$

Cube: $r \wedge q$ provides the models:

$i_4 : \{p,q,r\} \to \{T,F\}, i_4(p) = T, i_4(q) = T, i_4(r) = T$ and $i_4(r \wedge q) = T$

$i_5 : \{p,q,r\} \to \{T,F\}, i_5(p) = F, i_5(q) = T, i_5(r) = T$ and $i_5(r \wedge q) = T$

Note that $i_2 = i_5$, thus U has four distinct models i_1, i_2, i_3, i_4:

$i_1(U) = i_2(U) = i_3(U) = i_4(U) = T$

All the other four interpretations, from the total of eight possible interpretations, evaluate the formula U as false, they are anti-models of U.

Example 1.8.

Using the conjunctive normal form of $U = p \vee q \to p \wedge q$ find all its anti-models.

The normalization algorithm is applied in order to obtain CNF(U).

$U = p \vee q \to p \wedge q \equiv \neg(p \vee q) \vee (p \wedge q) \equiv (\neg p \wedge \neg q) \vee (p \wedge q)$ apply distributivity

$\equiv (\neg p \vee p) \wedge (\neg p \vee q) \wedge (\neg q \vee p) \wedge (\neg q \vee q) \equiv$

$\equiv T \wedge (\neg p \vee q) \wedge (\neg q \vee p) \wedge T \equiv (\neg p \vee q) \wedge (\neg q \vee p)$ - CNF with 2 clauses

The anti-models of U are the interpretations which evaluate one by one the clauses of CNF as false. In a clause we assign the truth value F to all its literals.

Clause: $\neg p \vee q$ provides the anti-model:

$i_1 : \{p,q\} \to \{T,F\}, i_1(p) = T, i_1(q) = F$, thus $i_1(\neg p \wedge q) = F$ and $i_1(U) = F$

Clause: $\neg q \vee p$ provides the anti-model:

$i_2 : \{p,q\} \to \{T,F\}, i_2(p) = F, i_2(q) = T$ thus $i_2(\neg q \vee p) = F$ and $i_2(U) = F$

U has two anti-models: i_1 and i_2, which evaluate the formula as false. The total number of its interpretations is $2^2 = 4$, therefore U has two models which evaluate the formula as true, and they correspond to the other two possible assignments of T and F to the variables p and q.

1.5. *Formal (axiomatic) system of propositional logic*

In this section we present the formalization of propositional logic proposed in paper [62]. From a syntactic perspective propositional logic is defined using the axiomatic system: $P=(\Sigma_P, F_P, A_P, R_P)$ where:

1. $\Sigma_P = Var\_propos \cup \{T,F\} \cup Connectives \cup \{(,)\}$ is the ***vocabulary***
   - $Var\_propos = \{p_1, p_2, ...\}$ is the ***set*** of ***propositional variables***
   - $Connectives = \{\neg(negation), \wedge(conjunction), \vee(disjunction),$ $\rightarrow(implication), \leftrightarrow(equivalence)\}$ - the set of ***logical connectives***
2. F_P is the ***set of well-formed formulas***, which is the smallest set of formulas satisfying the rules:
   - *base*: $p_i \in F_P, i=1,2,...;$
   - *induction*:
     if $U,V \in F_P$ then:
     $$\neg U \in F_P, U \wedge V \in F_P, U \vee V \in F_P, U \rightarrow V \in F_P, U \leftrightarrow V \in F_P$$
   - *closure*: all formulas from F_P are obtained by applying the above rules in a finite number of steps.
3. $A_P = \{A_1, A_2, A_3\}$ is the set of axioms of propositional logic
   $A_1 : U \rightarrow (V \rightarrow U)$
   $A_2 : (U \rightarrow (V \rightarrow Z)) \rightarrow ((U \rightarrow V) \rightarrow (U \rightarrow Z))$
   $A_3 : (U \rightarrow V) \rightarrow (\neg V \rightarrow \neg U)$ - *modus tollens*
   These axioms are in fact axiomatic schemes, where U,V,Z are arbitrary formulas, generating an infinite number of axioms.
4. $R_P = \{mp\}$ is the set of ***inference*** (***deduction***) ***rules*** containing ***modus ponens rule***. Notation: $U, U \rightarrow V \vdash_{mp} V$, with the meaning: "from the formulas U and $U \rightarrow V$ we deduce (infer) V".

Remark:

- All the axioms obtained using the axiomatic schemes A_1, A_2, A_3 are tautologies (valid formulas).

Definition 1.8.

Let $U_1, U_2, ..., U_n$ be formulas, called ***hypotheses*** and V be a formula called ***conclusion.*** V ***is deducible (derivable, inferable) from*** $U_1, U_2, ..., U_n$ and we denote this by $U_1, U_2, ..., U_n \vdash V$, if there is a sequence $(f_1, f_2, ..., f_m)$ of formulas such that $f_m = V$ and $\forall i \in \{1, ..., m\}$ we have a) or b) or c).

a) $f_i \in A_P$ (axiom);

b) $f_i \in \{U_1, U_2, ..., U_n\}$ (hypothesis formula);

c) $f_{i_1}, f_{i_2} \vdash_{mp} f_i$, $i_1 < i$ *and* $i_2 < i$ (formula f_i is inferred using *modus ponens* from two existing formulas)

The sequence $(f_1, f_2, ..., f_m)$ is called the ***deduction of*** V ***from*** the hypotheses $U_1, U_2, ..., U_n$.

Definition 1.9.
A formula $U \in F_P$, such that $\varnothing \vdash U$ (or $\vdash U$) is called ***theorem.***

Remark: The theorems are the formulas deducible (inferable) only from the axioms and using *modus ponens* as inference rule.

This axiomatic system (P) belongs to the group of Hilbert axiomatic systems [24], all of them having only *modus ponens* as inference rule. The formal system (P) presented above has as a basic binary connective *implication* ($\rightarrow$). Other Hilbert systems change the axioms or use also *disjunction* ($\vee$) as a basic connective. All these axiomatic systems are equivalent: the axioms of such a system can be proved as theorems in all the other axiomatic systems.

Example 1.9.
Prove that: $U, U \rightarrow V, V \rightarrow Z \vdash Z$ using the definition of deduction. $U, V, Z \in F_P$

We build the sequence $(f_1, f_2, f_3, f_4, f_5)$ of formulas as follows:

$f_1 : U$

$f_2 : U \rightarrow V$

$f_1, f_2 \vdash_{mp} V$

$f_3 : V$

$f_4 : V \rightarrow Z$

$f_3, f_4 \vdash_{mp} Z$

$f_5 : Z$

Note that f_1, f_2, f_4 are initial formulas (hypotheses), and f_3, f_5 are obtained using *modus ponens* rule. According to Definition 1.8 the sequence: $(f_1, f_2, f_3, f_4, f_5)$ is the deduction of Z from the hypotheses $U, U \rightarrow V, V \rightarrow Z$.

Example 1.10.
Prove that: $\neg p \vee q, p \vee r, \neg q \vdash r$ using the definition of deduction.

In order to apply *modus ponens* the '$\vee$' connective must be written in an equivalent form using '$\rightarrow$'.

The sequence $(f_1, f_2, f_3, f_4, f_5, f_6, f_7)$ of formulas is built as follows:

$f_1 : \neg p \vee q \equiv p \to q$ (hypothesis)

$f_2 : p \vee r \equiv \neg p \to r$ (hypothesis)

$f_3 : \neg q$ (hypothesis)

$f_4 : (p \to q) \to (\neg q \to \neg p)$ - axiom A_3 (*modus tollens*)

$f_1, f_4 \vdash_{mp} \neg q \to \neg p$

$f_5 : \neg q \to \neg p$

$f_3, f_5 \vdash_{mp} \neg p$

$f_6 : \neg p$

$f_2, f_6 \vdash_{mp} r$

$f_7 : r$ (conclusion)

Using only the hypotheses we cannot prove the conclusion, so we need to use axioms. In the deduction process modelled by the sequence $(f_1, f_2, f_3, f_4, f_5, f_6, f_7)$, the formulas f_1, f_2, f_3 are hypotheses, f_4 is an instantiation of axiom A_3 and f_5, f_6, f_7 are derived using *modus ponens*. According to Definition 1.8, we have proved that the deduction holds.

Example 1.11.

Prove that $\vdash U \wedge V \to V$ using the definition of deduction.

We transform syntactically the formula $U \wedge V \to V$ such that the only connectives are $\neg, \to$.

The conjunction is replaced using the following logical equivalence:

$U \wedge V \equiv \neg(\neg U \vee \neg V) \equiv \neg(U \to \neg V)$

$U \wedge V \to V \equiv \neg(U \to \neg V) \to V$

We use the axioms A_1 and A_3 as follows:

$f_1 : \neg V \to (U \to \neg V)$ - an instance of axiom A_1

$f_2 : (\neg V \to (U \to \neg V)) \to (\neg(U \to \neg V) \to V)$ - axiom A_3 (*modus tollens*)

$f_1, f_2 \vdash_{mp} \neg(U \to \neg V) \to V \equiv \neg(\neg U \vee \neg V) \to V \equiv U \wedge V \to V$

The formula $U \wedge V \to V$ was deduced only from the axioms and *modus ponens* inference rule, therefore it is a theorem.

Example 1.12.

Prove that $\vdash U \to U$, building the deduction of $U \to U$ from the axioms only.

The sequence of formulas $(f_1, f_2, f_3, f_4, f_5)$ is built.

$f_1 : U \to ((U \to U) \to U)$ obtained from $A_1 : U \to (V \to U)$

replacing V by $U \to U$;

$f_2 : (U \to ((U \to U) \to U)) \to ((U \to (U \to U)) \to (U \to U))$ obtained
from $A_2 : (U \to (V \to Z)) \to ((U \to V) \to (U \to Z))$
using the replacements: V by $U \to U$ and Z by U;

$f_1, f_2 \vdash_{mp} (U \to (U \to U)) \to (U \to U)$

$f_3 : (U \to (U \to U)) \to (U \to U)$

$f_4 : U \to (U \to U)$ obtained from $A_1 : U \to (V \to U)$ replacing V by U

$f_3, f_4 \vdash_{mp} U \to U$

$f_5 : U \to U$

Therefore the sequence $(f_1, f_2, f_3, f_4, f_5)$ is the deduction of $U \to U$ from the axioms, so $U \to U$ is a theorem.

Example 1.13.
Prove that $\neg U \vdash U \to V$ using the definition of deduction.

In the inferential process the sequence of formulas $(f_1, f_2, f_3, f_4, f_5)$ is built as follows:

$f_1 : \neg U$ – hypothesis formula

$f_2 : \neg U \to (\neg V \to \neg U)$ obtained from $A_1 : U \to (V \to U)$
replacing U by $\neg U$ and V by $\neg V$;

$f_1, f_2 \vdash_{mp} \neg V \to \neg U$

$f_3 : \neg V \to \neg U$

$f_4 : (\neg V \to \neg U) \to (U \to V)$ obtained from $A_3 : (U \to V) \to (\neg V \to \neg U)$
replacing U by $\neg V$ and V by $\neg U$;

$f_3, f_4 \vdash_{mp} U \to V$

$f_5 : U \to V$

The sequence $(f_1, f_2, f_3, f_4, f_5)$ is the deduction of $U \to V$ from $\neg U$ and the axioms.

In the inferential (deductive) process, beside *modus ponens*, other inference rules are used. These rules can be also expressed as theorems.

addition	$U \vdash U \vee V$	inference rule
	$\vdash U \to U \vee V$	theorem
simplification	$U \wedge V \vdash U$	inference rule
	$\vdash U \wedge V \to U$	theorem
	$U \wedge V \vdash V$	inference rule
	$\vdash U \wedge V \to V$	theorem

conjunction	$U,V \vdash U \wedge V$	inference rule
modus ponens	$U,U \to V \vdash V$	inference rule
	$\vdash U \wedge (U \to V) \to V$	theorem
modus tollens	$\neg V,U \to V \vdash \neg U$	inference rule
	$\vdash \neg V \wedge (U \to V) \to \neg U$	theorem
	$U \to V \vdash \neg V \to \neg U$	inference rule
	$\vdash (U \to V) \to (\neg V \to \neg U)$	theorem
syllogism	$U \to V, V \to Z \vdash U \to Z$	inference rule
	$\vdash (U \to V) \wedge (V \to Z) \to (U \to Z)$	theorem
	$U,U \to V, V \to Z \vdash Z$	inference rule
	$\vdash U \wedge (U \to V) \wedge (V \to Z) \to Z$	theorem
resolution	$U \vee V, \neg U \vee Z \vdash V \vee Z$	inference rule
	$\vdash (U \vee V) \wedge (\neg U \vee Z) \to (V \vee Z)$	theorem

1.6. The theorem of deduction and its reverse

Some theorems, even if they are syntactically simple, are very difficult to be proved using only the axioms and *modus ponens*, because we need to guess the starting-point axiomatic schemes and the corresponding replacements of the arbitrary formulas U,V,Z .

The combination of the following two theoretic results is used to help us prove at the syntactic level theorems having a special syntactic form (the connectives are mainly ' $\to$ ').

Theorem 1.6. Theorem of deduction:
If $U_1,...,U_{n-1},U_n \vdash V$, then $U_1,...,U_{n-1} \vdash U_n \to V$.

Theorem 1.7. Reverse of the theorem of deduction:
If $U_1,...,U_{n-1} \vdash U_n \to V$ then $U_1,...,U_{n-1},U_n \vdash V$.

Applying n times the theorem of deduction and its reverse we obtain:

$U_1,...,U_{n-1},U_n \vdash V$ if and only if

$U_1,...,U_{n-1} \vdash U_n \to V$ if and only if

$U_1,...,U_{n-2} \vdash U_{n-1} \to (U_n \to V)$ if and only if

$U_1 \vdash U_2 \to (... \to (U_{n-1} \to (U_n \to V))...)$ if and only if

$\vdash U_1 \to (U_2 \to (... \to (U_n \to V)...))$

Note: The application of the theorem of deduction and its reverse means that a premise (hypothesis) of a deduction can be moved from the left-hand side of '$\vdash$' to its right-hand side as a premise of a main '$\rightarrow$' and vice-versa.

Consequences of the theorem of deduction:

1. $\vdash U \rightarrow ((U \rightarrow V) \rightarrow V)$;
2. $\vdash (U \rightarrow V) \rightarrow ((V \rightarrow Z) \rightarrow (U \rightarrow Z))$ the *syllogism* law;
3. $\vdash (U \rightarrow (V \rightarrow Z)) \rightarrow (V \rightarrow (U \rightarrow Z))$ the *permutation of the premises* law;
4. $\vdash (U \rightarrow (V \rightarrow Z)) \rightarrow (U \wedge V \rightarrow Z))$ the *reunion of the premises* law;
5. $\vdash (U \wedge V \rightarrow Z) \rightarrow (U \rightarrow (V \rightarrow Z))$ the *separation of the premises* law.

A syntactic and direct proof method based on these theorems consists of three steps:

Step 1: The initial theorem (formula) to be proved is written in an equivalent form as a deduction, applying the reverse of the theorem of deduction several times.

Step 2: The deduction obtained in Step1 is proved using the axiomatic system.

Step 3: Starting with the deduction proved in Step 2 we apply the theorem of deduction several times. All the premises moved in Step 1 from right to left are now moved from left to right in the reverse order to obtain the initial formula.

Remarks:

- Step1 may be skipped if we can guess the starting-point deduction in order to prove the initial formula.
- In Step 3, if we change the order in which the premises of the deduction are moved from left to right, new theorems can be proved.

Example 1.14.

The consequences 1 and 2 are proved in the following.

1. We begin with the *modus ponens* rule: $U, U \rightarrow V \vdash_{mp} V$

   Applying the theorem of deduction we obtain: $U \vdash (U \rightarrow V) \rightarrow V$ and we continue with another application of the same theorem: $\vdash U \rightarrow ((U \rightarrow V) \rightarrow V)$.

2. We begin with the deduction: $U, U \rightarrow V, V \rightarrow Z \vdash Z$ proved in Example 1.9.

   Application of the theorem of deduction (U is moved from left to right):

   $$U \rightarrow V, V \rightarrow Z \vdash U \rightarrow Z$$

   Application of the theorem of deduction ($V \rightarrow Z$ is moved from left to right):

   $$U \rightarrow V \vdash (V \rightarrow Z) \rightarrow (U \rightarrow Z)$$

   Application of the theorem of deduction ($U \rightarrow V$ is moved from left to right):

   $$\vdash (U \rightarrow V) \rightarrow ((V \rightarrow Z) \rightarrow (U \rightarrow Z))$$

Example 1.15.
Using the theorem of deduction and its reverse prove:
$\vdash (p \to r) \to ((p \wedge r \to q) \to (p \to q))$

Step 1:
The reverse of the theorem of deduction is applied to obtain the starting-point deduction:
if $\vdash (p \to r) \to ((p \wedge r \to q) \to (p \to q))$ then
$p \to r \vdash (p \wedge r \to q) \to (p \to q)$ then
$p \to r, p \wedge r \to q \vdash p \to q$ then $p \to r, p \wedge r \to q, p \vdash q$

Step 2:
We prove the deduction obtained in Step 1.
Using Definition 1.8 we build the sequence of formulas: $(f_1, f_2, f_3, f_4, f_5, f_6)$:

$f_1 : p$ – premise (hypothesis)

$f_2 : p \to r$ – premise

$f_1, f_2 \vdash_{mp} r$

$f_3 : r$

$f_4 : f_1 \wedge f_3 = p \wedge r$ (conjunction of the conclusions)

$f_5 : p \wedge r \to q$ – premise

$f_4, f_5 \vdash_{mp} q$

$f_6 : q$

The sequence $(f_1, f_2, f_3, f_4, f_5, f_6)$ is the deduction of q from the premises: $p \to r, p \wedge r \to q, p$.

Step 3:
To the deduction $p \to r, p \wedge r \to q, p \vdash q$ we apply three times the theorem of deduction.
There are $3! = 6$ such possibilities (to move the premises to the right-hand side of the meta-symbol $\vdash$) and we prove 6 theorems: $T_1, T_2, T_3, T_4, T_5, T_6$

1. *the premises are moved to the right-hand side of '$\vdash$' in the following order:*

p, $p \wedge r \to q$, $p \to r$.

if $p \to r, p \wedge r \to q, p \vdash q$ then

$p \to r, p \wedge r \to q \vdash p \to q$ then

$p \to r \vdash (p \wedge r \to q) \to (p \to q)$ then

$\vdash T_1 = (p \to r) \to ((p \wedge r \to q) \to (p \to q))$ – the theorem to be proved.

2. *the premises are moved to the right-hand side of '$\vdash$' in the following order:*

$p,\ p \to r,\ p \wedge r \to q$.

if $p \to r, p \wedge r \to q, p \vdash q$ then

$p \to r, p \wedge r \to q \vdash p \to q$ then

$p \wedge r \to q \vdash (p \to r) \to (p \to q)$ then

$\vdash T_2 = (p \wedge r \to q) \to ((p \to r) \to (p \to q))$

The following theorems can be also proved:

$\vdash T_3 = (p \wedge r \to q) \to (p \to ((p \to r) \to q)))$

$\vdash T_4 = p \to ((p \wedge r \to q) \to ((p \to r) \to q))$

$\vdash T_5 = (p \to r) \to (p \to ((p \wedge r \to q) \to q))$

$\vdash T_6 = p \to ((p \to r) \to ((p \wedge r \to q) \to q))$

1.7. Properties of propositional logic

The properties of propositional logic are: ***compactness, soundness, completeness, coherence, non-contradiction and decidability.***

Theorem 1.8. (compactness 1)

An infinite set of propositional formulas has a model if and only if each of its subsets has a finite model.

Theorem 1.9.

Let $S = \{U_1, U_2, \ldots, U_m, \ldots\}$ be an infinite set of propositional formulas.

1. *S is inconsistent **if and only** if $\exists k \in N^*$, such that $\{U_1, U_2, \ldots, U_k\}$ is inconsistent.*
2. *S is consistent **if and only if***

   *$\{U_1\}$ is consistent **and***

   *$\{U_1, U_2\}$ is consistent **and***

   ...

   *$\{U_1, U_2, \ldots, U_m\}$ is consistent **and***

   ...

Remarks:

- An infinite set of propositional formulas is inconsistent if and only if it has an inconsistent finite subset, therefore *inconsistency can be proved in a finite number of steps.*
- An infinite set of propositional formulas is consistent if and only if all its subsets (an infinite number) are consistent, therefore *consistency cannot be proved in a finite number of steps.*

Theorem 1.10. (compactness 2)
A propositional formula V is a logical consequence of an infinite set of propositional formulas S (notation: $S \models V$) if and only if there is a finite subset $\{U_1, U_2, ..., U_n\} \subset S$ such that $U_1, U_2, ..., U_n \models V$.

Theorem 1.11. Soundness theorem
(syntactic validity implies semantic validity):
If $\vdash U$ then $\models U$ (a theorem is a tautology).

Theorem 1.12. Completeness theorem
(semantic validity implies syntactic validity):
If $\models U$ then $\vdash U$ (a tautology is a theorem).

Theorem 1.13. Soundness and completeness for propositional logic:
$\vdash U$ if and only if $\models U$ (a formula is a theorem if and only if it is a tautology).

Consequences of the last theorem are the following properties:
1. ***Propositional logic is non-contradictory***: we cannot have simultaneously: $\vdash U$ and $\vdash \neg U$.
2. ***Propositional logic is coherent***: not every propositional formula is a theorem.
3. ***Propositional logic is decidable***: always we can decide if a propositional formula is a theorem or not. The truth table method is a decision method.

1.8. Decision problems and proof methods

Decision problems in propositional logic:
1. Is a propositional formula a tautology/theorem?
   $\overset{?}{\models} V$ or $\overset{?}{\vdash} V$
2. Is a propositional formula a logical/syntactic consequence of a set of hypotheses?
   $U_1, ..., U_n \overset{?}{\models} V$ or $U_1, ..., U_n \overset{?}{\vdash} V$

In order to solve these two decision problems, theorem proving methods are applied. These methods will be studied in the next chapters.
In the following we provide w the classifications of the studied proof methods from two perspectives:

First classification: **semantic** *versus* **syntactic methods**
1. ***semantic proof methods***:
   - the truth table method;
   - the semantic tableaux method;
   - CNF- conjunctive normal form.

2. ***syntactic proof methods***:
   - the definition of deduction;
   - the theorem of deduction and its reverse;
   - the resolution method;
   - the sequent calculus method.

Second classification: **direct** *versus* **refutation methods**

1. ***direct methods***: they use directly the formula to be proved:
   - the truth table method;
   - the CNF- conjunctive normal form;
   - the definition of deduction;
   - the theorem of deduction and its reverse;
   - the sequent calculus method.

2. ***refutation methods***: they model the "reductio ad absurdum" (proof by contradiction) using the negation of the formula to be proved:
   - the semantic tableaux method;
   - the resolution method.

1.9. Exercises

Exercise 1.1.

Check the following properties for $\downarrow$ ('*nor*'), $\uparrow$ ('*nand*') and $\oplus$ ('*xor*') connectives using the truth table method.

1. associativity of '$\uparrow$' connective:
   $$p \uparrow (q \uparrow r) \equiv (p \uparrow q) \uparrow r ;$$
2. associativity of '$\downarrow$' connective:
   $$p \downarrow (q \downarrow r) \equiv (p \downarrow q) \downarrow r ;$$
3. associativity of '$\oplus$' connective:
   $$p \oplus (q \oplus r) \equiv (p \oplus q) \oplus r ;$$
4. distribution of '$\uparrow$' connective over '$\downarrow$' connective:
   $$p \uparrow (q \downarrow r) \equiv (p \uparrow q) \downarrow (p \uparrow r) ;$$
5. distribution of '$\downarrow$' connective over '$\uparrow$' connective:
   $$p \downarrow (q \uparrow r) \equiv (p \downarrow q) \uparrow (p \downarrow r) ;$$
6. De Morgan's laws for '$\downarrow$' and '$\uparrow$':
   $$\neg(p \downarrow q) \equiv \neg p \uparrow \neg q \quad \text{and} \quad \neg(p \uparrow q) \equiv \neg p \downarrow \neg q ;$$
7. $p \uparrow (q \vee r) \equiv (p \uparrow q) \wedge (p \uparrow r)$ and $p \downarrow (q \wedge r) \equiv (p \downarrow q) \vee (p \downarrow r)$.
8. $p \downarrow (q \uparrow p) \equiv F$ and $p \uparrow (q \downarrow p) \equiv T ;$

Exercise 1.2.
Using the truth table method decide what kind of formula (consistent, inconsistent, tautology, contingent) is $U_j, j \in \{1,2,...,8\}$. Write all the models and anti-models of $U_j, j \in \{1,2,...,8\}$.

1. $U_1 = q \wedge \neg p \wedge r \rightarrow \neg p \vee \neg(q \wedge r)$;
2. $U_2 = \neg p \vee \neg(q \wedge r) \rightarrow q \wedge \neg p$;
3. $U_3 = \neg p \wedge (\neg q \vee r) \rightarrow q \vee \neg p \vee r$;
4. $U_4 = \neg(\neg p \vee q) \vee r \rightarrow \neg p \vee (\neg q \vee r)$;
5. $U_5 = \neg p \vee (\neg q \vee \neg r) \rightarrow q \wedge \neg p$;
6. $U_6 = \neg p \vee (\neg q \wedge \neg r) \rightarrow q \wedge \neg p \wedge r$;
7. $U_7 = p \rightarrow (q \wedge r) \vee q \wedge \neg p$;
8. $U_8 = (p \vee q) \wedge \neg r \rightarrow p \wedge q \wedge r$.

Exercise 1.3.
Using the truth table method, check if the following logical consequences hold:

1. $p \rightarrow q \models (p \rightarrow r) \rightarrow (p \rightarrow q \wedge r)$;
2. $p \rightarrow q \models (q \rightarrow r) \rightarrow (p \rightarrow r)$;
3. $p \rightarrow (q \rightarrow r) \models (p \rightarrow q) \rightarrow (p \rightarrow r)$;
4. $p \rightarrow r \models (q \rightarrow r) \rightarrow ((p \vee q) \rightarrow r)$;
5. $p \rightarrow q \models (\neg p \rightarrow q) \rightarrow q$;
6. $p \rightarrow q \models (q \rightarrow r) \rightarrow (p \rightarrow q \wedge r)$;
7. $p \rightarrow q \models (q \rightarrow r) \rightarrow (p \rightarrow q \vee r)$;
8. $r \rightarrow (q \rightarrow p) \models (r \rightarrow q) \rightarrow (r \rightarrow p)$.

Exercise 1.4.
Prove that the following formulas are tautologies using the truth table method.

1. the left-distribution of ' $\rightarrow$ ' over ' $\wedge$ ': $(p \rightarrow (q \wedge r)) \rightarrow ((p \rightarrow q) \wedge (p \rightarrow r))$;
2. the permutation of the premises law: $(p \rightarrow (q \rightarrow r)) \rightarrow (q \rightarrow (p \rightarrow r))$;
3. the reunion of the premises law: $(p \rightarrow (q \rightarrow r)) \rightarrow (p \wedge q \rightarrow r)$;
4. the separation of the premises law: $(p \wedge q \rightarrow r) \rightarrow (p \rightarrow (q \rightarrow r))$;
5. the 'cut' law: $(p \rightarrow q) \wedge (p \wedge q \rightarrow r) \rightarrow (p \rightarrow r)$;
6. the left-distribution of ' $\vee$ ' over ' $\rightarrow$ ': $p \vee (q \rightarrow r) \rightarrow ((p \vee q) \rightarrow (p \vee r))$;
7. the syllogism law: $(p \rightarrow q) \wedge (q \rightarrow r) \rightarrow (p \rightarrow r)$;
8. the left-distribution of ' $\rightarrow$ ' over ' $\vee$ ': $(p \rightarrow (q \vee r)) \rightarrow ((p \rightarrow q) \vee (p \rightarrow r))$.

Exercise 1.5.
Transform the formulas $U_j, j \in \{1,2,...,8\}$ into their equivalent conjunctive and disjunctive normal forms. Using one of these forms prove that $U_j, j \in \{1,2,...,8\}$ are valid formulas in propositional logic.

1. $U_1 = (p \rightarrow (q \leftrightarrow r)) \rightarrow ((p \rightarrow q) \leftrightarrow (p \rightarrow r))$;
2. $U_2 = (p \rightarrow q) \wedge (p \wedge q \rightarrow r) \rightarrow (p \rightarrow r)$;
3. $U_3 = (p \wedge q \rightarrow r) \rightarrow (p \rightarrow (q \rightarrow r))$;

4. $U_4 = (p \to (q \vee r)) \to ((p \to q) \vee (p \to r))$;
5. $U_5 = (p \vee (q \leftrightarrow r)) \to ((p \vee q) \leftrightarrow (p \vee r))$;
6. $U_6 = (p \to (q \to r)) \to ((p \to q) \to (p \to r))$;
7. $U_7 = (p \to (q \wedge r)) \to ((p \to q) \wedge (p \to r))$;
8. $U_8 = p \vee (q \to r) \to ((p \vee q) \to (p \vee r))$.

Exercise 1.6.
Using the appropriate normal form write all the models of the following formulas:

1. $U_1 = (p \vee q \to r) \to (p \to r) \wedge q$;
2. $U_2 = \neg(\neg p \vee q) \vee r \to \neg p \wedge \neg(q \wedge r)$;
3. $U_3 = (p \wedge q \to r) \to (p \to r) \wedge q$;
4. $U_4 = (p \vee q) \wedge \neg r \to p \wedge q \wedge r$;
5. $U_5 = p \vee \neg(q \wedge \neg r) \to p \wedge q \wedge \neg r$;
6. $U_6 = (p \vee q \to r) \to (q \to r) \wedge p$;
7. $U_7 = (q \vee r \to p) \to (p \to r) \wedge q$;
8. $U_8 = (q \wedge r \to p) \to (p \to r) \wedge q$.

Exercise 1.7.
Using the appropriate normal form, prove that the following formulas are inconsistent:

1. $U_1 = (p \to (q \to r)) \wedge \neg((p \to q) \to (p \to r))$;
2. $U_2 = (\neg p \vee q) \wedge \neg(\neg q \to \neg p)$;
3. $U_3 = (p \to q) \wedge (p \wedge q \to r) \wedge (p \wedge \neg r)$;
4. $U_4 = (p \to (q \vee r)) \wedge (\neg(p \to q) \wedge \neg(p \to r))$;
5. $U_5 = p \wedge (q \to r) \wedge ((p \wedge q) \wedge \neg(p \wedge r))$;
6. $U_6 = (p \to (q \to r)) \wedge (p \wedge q \wedge \neg r)$;
7. $U_7 = (p \to (q \to r)) \wedge \neg(q \to (p \to r))$;
8. $U_8 = (p \wedge q \to r) \wedge \neg(p \to (q \to r))$.

Exercise 1.8.
Write all the anti-models of the following formulas using CNF.

1. $U_1 = (q \wedge r \to p) \to (p \to r) \wedge q$;
2. $U_2 = (q \vee r \to p) \to (p \to r) \wedge q$;
3. $U_3 = (p \vee q \to r) \to (q \to r) \wedge p$;
4. $U_4 = p \vee \neg(q \wedge \neg r) \to p \wedge q \wedge \neg r$;
5. $U_5 = p \vee \neg(q \wedge \neg r) \to p \wedge q \wedge \neg r$;

6. $U_6 = (p \wedge q \to r) \to (p \to r) \wedge q$;
7. $U_7 = \neg(\neg p \vee q) \vee r \to \neg p \wedge \neg(q \wedge r)$;
8. $U_8 = (p \vee q \to r) \to (p \to r) \wedge q$.

Exercise 1.9.
Using the definition of deduction, prove the following deductions:

1. $p \to q, r \to t, p \vee r, \neg q \vdash t$;
2. $p \to r, p \vee r \to q, r \vdash q$;
3. $q \to p, t \to r, q \vee t, \neg p \vdash r$;
4. $p \vee (q \to r), p \vee q, \neg p \vdash r$;
5. $\neg p \vee \neg q \vee r, q, p \vdash r$;
6. $p \to \neg q \vee r, p \wedge q, p \vdash r$;
7. $r \vee (q \to p), r \vee q, \neg r \vdash p$;
8. $p \to q, q \to r, r \to t, p \vdash t$.

Exercise 1.10.
Prove the following theorems using the theorem of deduction and its reverse.

1. $\vdash p \vee (q \to r) \to ((p \vee q) \to (p \vee r))$;
2. $\vdash (p \to (\neg r \to q)) \to (r \vee \neg p \vee q)$;
3. $\vdash (p \to (q \to r) \to (p \wedge q \to r)$;
4. $\vdash (p \wedge q \to r) \to (p \to (q \to r))$;
5. $\vdash (p \to (q \to r)) \to (q \to (p \to r))$;
6. $\vdash (p \to (q \to r)) \to ((p \to q) \to (p \to r))$;
7. $\vdash (p \to q) \wedge (p \wedge q \to r) \to (p \to r)$;
8. $\vdash (p \to q) \to ((p \to r) \to (p \to q \wedge r))$.

Exercise 1.11.
Using the theorem of deduction and its reverse prove that:

1. $\vdash (p \to (q \vee r)) \to ((p \to q) \vee (p \to r))$;
2. $\vdash (p \to q) \to ((\neg r \vee p) \to (r \to q))$;
3. $\vdash p \vee (q \to r) \to ((p \vee q) \to (p \vee r))$;
4. $\vdash (p \to r) \to ((q \to r) \to (p \vee q \to r))$;
5. $\vdash (p \to q) \to ((r \to t) \to (p \wedge r \to q \wedge t))$;
6. $\vdash (p \to r) \to ((p \wedge r \to q) \to (p \to q))$;
7. $\vdash (\neg q \vee p) \to ((s \to q) \to (s \to p))$;
8. $\vdash (p \to (q \to r)) \to (p \to (\neg r \to \neg q))$.

2. FIRST-ORDER LOGIC

First-order (predicate) logic was introduced by Gottlob Frege in 1879. It is an extension of propositional logic and allows reasoning about the objects of some non-empty universe and the relations among these objects. As references for the theoretical concepts and results presented in this chapter, the following papers were used [1, 2, 12, 16, 19, 20, 21, 23, 26, 37, 43, 44, 47, 54, 58, 60, 61, 62, 63].

2.1. The axiomatic (formal) system of first-order logic

In predicate logic new syntactic categories: quantifiers (existential and universal), constants, variables, function symbols and predicate symbols, are introduced.
We present an axiomatic (deductive) system for predicate logic as was proposed in paper [62]: $\mathrm{Pr} = (\Sigma_{\mathrm{Pr}}, F_{\mathrm{Pr}}, A_{\mathrm{Pr}}, R_{\mathrm{Pr}})$, where:

- $\Sigma_{\mathrm{Pr}} = Var \cup Const \cup (\bigcup_{j=1}^{n} \mathrm{F}_j) \cup (\bigcup_{j=1}^{m} \mathrm{P}_j) \cup Connectives \cup Quantifiers$ is the ***vocabulary***
  - ***Var*** is the set of variable symbols $\{x, y, z, \ldots\}$; the variables take different values in a specific domain D and they are generic terms, type definitions: ***book, child, event.***
  - ***Const*** is the set of constants $\{a, b, c, \ldots\}$; the constants take fixed values in a domain D and they usually specify objects names, persons names: ***Paul, book_3.***
  - $\mathrm{F}_i = \{f \mid f : D^i \to D\}$ is the set of function symbols of arity "i"
  - $\mathrm{P}_i = \{p \mid p : D^i \to \{T, F\}\}$ is the set of predicate symbols of arity "i" and usually represent connection rules among variables and constants.
  - ***Connectives*** $= \{\neg, \wedge, \vee, \to, \leftrightarrow\}$;
  - ***Quantifiers*** = { $\forall$ (*universal quantifier*), $\exists$ (*existential quantifier*)}
    In first-order logic quantifiers refer only variables. The scope of a quantifier is the formula to which the quantifier applies. The priority of the quantifiers is greater than the priorities of the connectives.

  To define the well-formed predicate formulas we need to introduce first the concepts: term, atom, literal.

- ***TERMS*** is the set of terms defined as follows:
  - $Var \subset TERMS$; $Const \subset TERMS$;
  - if $f \in \mathrm{F}_k$ and $t_1, t_2, \ldots, t_k \in TERMS$ then $f(t_1, t_2, \ldots, t_k) \in TERMS$
  - examples: x, a, $f(x)$, $g(x, a)$, $g(f(x), y)$

- ***ATOMS*** is the set of atomic formulas (atoms):
  - $T, F \in ATOMS$
  - if $P \in \mathrm{P}_k$ and $t_1,...,t_k \in TERMS$ then $P(t_1,...,t_k) \in ATOMS$
  - examples: $T, F, P(x,y,a), Q(f(x),a), R(a,g(f(x),y)$
    where: $x, y \in Var, a \in Const$, $f \in \mathrm{F}_1, g \in \mathrm{F}_2$, $P, R \in \mathrm{P}_3, Q \in \mathrm{P}_2$.
- ***literal*** - an atom or its negation.
  - examples: $P(f(x),a,y,g(x,b))$, $\neg Q(x,a,f(x))$,
    where: $x, y \in Var$, $a, b \in Const$, $f \in \mathrm{F}_1, g \in \mathrm{F}_2, P \in \mathrm{P}_4, Q \in \mathrm{P}_3$.

- F_{Pr} is the set of well-formed predicate (first-order) formulas:
  - if $U, V \in F_{\mathrm{Pr}}$ then
    $\neg U \in F_{\mathrm{Pr}}$, $U \wedge V \in F_{\mathrm{Pr}}$, $U \vee V \in F_{\mathrm{Pr}}$, $U \rightarrow V \in F_{\mathrm{Pr}}$, $U \leftrightarrow V \in F_{\mathrm{Pr}}$
  - if $U \in F_{\mathrm{Pr}}$, $x \in Var$, and x is not within the scope of a quantifier then $(\forall x)U(x) \in F_{\mathrm{Pr}}$ and $(\exists x)U(x) \in F_{\mathrm{Pr}}$.
- $A_{\mathrm{Pr}} = \{A_1, A_2, A_3, A_4, A_5\}$ is the set of axioms:
  $A_1 : U \rightarrow (V \rightarrow U)$
  $A_2 : ((U \rightarrow (V \rightarrow Z)) \rightarrow ((U \rightarrow V) \rightarrow (U \rightarrow Z))$
  $A_3 : (U \rightarrow V) \rightarrow (\neg V \rightarrow \neg U)$ (*modus tollens*)
  $A_4 : (\forall x)U(x) \rightarrow U(t)$, where t is a term, (*universal instantiation*)
  $A_5 : (U \rightarrow V(y)) \rightarrow (U \rightarrow (\forall x)V(x))$, where x is not a free variable in U or V, y is free in V and y does not appear in U.
- $R_{\mathrm{Pr}} = \{mp, univ\_gen\}$ is the set of inference rules:
  - *modus ponens* rule: $U, U \rightarrow V \vdash_{mp} V$
  - *universal generalization* rule:
    $U(x) \vdash_{univ\_gen} (\forall x)U(x)$, where x is a free variable in U.

Remarks:

1. $A_P \subset A_{\mathrm{Pr}}$, $F_P \subset F_{\mathrm{Pr}}$, $R_P \subset R_{\mathrm{Pr}}$ and thus $Theorems_P \subset Theorems_{\mathrm{Pr}}$. The theorems in propositional logic are also theorems in predicate logic.
2. In axiom A_4 (*universal instantiation*), $(\forall x)U(x)$ says that $U(x)$ holds true for all the objects in the domain (universe), so U holds for any specific object in the domain. The term t used for instantiation can be a variable or a constant of the domain.
3. The *universal generalization* rule says that if U holds for any arbitrary element x of the universe, then we can conclude that $(\forall x)U(x)$.

Definition 2.1.

1. In a predicate formula the variables which are within the scope of a quantifier are called ***bound variables***, all the others are called ***free variables.***
2. A first-order formula is called a ***closed formula*** if all its variables are bound.
3. If a predicate formula contains at least one free variable, the ***formula is open***.

Example 2.1.

- The predicate formula $(\forall x)(\exists z)(P(x,z,a)\vee(\exists y)Q(x,f(y)))$ is closed (all variables are bound), where: $x,y,z\in Var$, $a\in Const$, $f\in \mathrm{F}_1$, $P\in \mathrm{P}_3$, $Q\in \mathrm{P}_2$.
- The predicate formula $(\forall x)P(x,y)\wedge Q(z,a)$ is open, $a\in Const$, $x,y,z\in Var$, $P,Q\in \mathrm{P}_2$, the variables y and z are free and x is a bound variable (universal quantified - within the scope of $\forall$).

Definition 2.2. [62]

Let $U_1,U_2,...,U_n,V$ be first-order formulas, $U_1,U_2,...,U_n$ are the hypotheses. V ***is deducible (inferable, derivable) from*** $U_1,U_2,...,U_n$, notation: $U_1,U_2,...,U_n\vdash V$, if there is a sequence of formulas $(f_1,f_2,...,f_m)$ such that $f_m=V$ and $\forall i\in\{1,...,m\}$ we have a) or b) or c) or d).

1. $f_i\in A_P$ (axiom of predicate logic);
2. $f_i\in\{U_1,U_2,...,U_n\}$ (hypothesis formula);
3. $f_{i_1},f_{i_2}\vdash_{mp} f_i, i_1<i \text{ and } i_2<i$ (formula f_i is inferred, using *modus ponens* rule, from two formulas that are already in the sequence);
4. $f_j\vdash_{univ\_gen} f_i, j<i$ (formula f_i is obtained using the *universal generalization* rule from a formula that exists already in the sequence).

The sequence $(f_1,f_2,...,f_m)$ is called the ***deduction*** of V from $U_1,U_2,...,U_n$.

Definition 2.3.

A formula $U\in F_{\mathrm{Pr}}$, such that $\varnothing\vdash U$ (notation: $\vdash U$) is called a ***theorem***.

Remark:

- The theorems are the formulas deducible from the axioms, using modus ponens and the universal generalization rule.

In the inferential (deduction) process the inference rules specific to propositional logic (*modus ponens, addition, simplification, modus tollens, syllogism*) and the rules specific to predicate logic (*universal instantiation, universal generalization, existential instantiation, existential generalization*) are used.

Inference rules for quantifiers	
universal instantiation	$(\forall x)U(x) \vdash_{univ\_inst} U(t)$, t is a term (variable, constant of the domain)
universal generalization	$U(x) \vdash_{univ\_gen} (\forall x)U(x)$, x is a free variable in U
existential instantiation	$(\exists x)U(x) \vdash_{exist\_inst} U(c)$, c is a constant of the domain
existential generalization	$U(t) \vdash_{exist\_gen} (\exists x)U(x)$, t is a variable or a constant of the domain, x must not appear free in U

Remarks:

1. The *existential instantiation rule* says that if U holds for some element of the universe, then we can give that element a name such as c. When selecting symbols, one must select them one at a time and must not use a symbol that has already been selected within the same reasoning/proof.
2. The *existential generalization rule* says that if there is some element t in the universe and t has the property U, then there exists something in the universe that has the property U.

Example 2.2.

Using the definition of deduction, prove that the formula $(\forall x)(P(x) \wedge Q(x)) \rightarrow (\forall x)P(x)$ is a theorem.

We build the sequence $(f_1, f_2, f_3, f_4, f_5)$ of formulas as follows:

$f_1 : (\forall x)(P(x) \wedge Q(x)) \rightarrow P(y) \wedge Q(y)$ - A_4 axiom where y is the term used for the *instantiation of the universal* quantified variable x

$f_2 : P(y) \wedge Q(y) \rightarrow P(y)$ – theorem (from propositional logic)

$f_1, f_2 \vdash_{syllogism\_rule} f_3 = (\forall x)(P(x) \wedge Q(x)) \rightarrow P(y)$

$f_4 : ((\forall x)(P(x) \wedge Q(x)) \rightarrow P(y)) \rightarrow ((\forall x)(P(x) \wedge Q(x)) \rightarrow (\forall x)P(x))$ - A_5 axiom

$f_3, f_4 \vdash_{mp} f_5 = (\forall x)(P(x) \wedge Q(x)) \rightarrow (\forall x)P(x)$ *modus ponens* is applied

$(f_1, f_2, f_3, f_4, f_5)$ is the deduction (proof) of the theorem $(\forall x)(P(x) \wedge Q(x)) \rightarrow (\forall x)P(x)$.

Example 2.3.

Using the definition of deduction, prove that:

$(\forall y)(\forall z)(P(y) \vee Q(z)) \vdash (\forall x)(P(x) \vee Q(x))$

The sequence $(f_1, f_2, f_3, f_4, f_5, f_6)$ of predicate formulas is obtained:

$f_1 : (\forall y)(\forall z)(P(y) \vee Q(z))$ – hypothesis

$f_2 : (\forall y)(\forall z)(P(y) \vee Q(z)) \rightarrow (\forall z)(P(x) \vee Q(z))$ - A_4 axiom where x is the term used for the *instantiation of the universal* quantified variable y

$f_1, f_2 \vdash_{mp} f_3 = (\forall z)(P(x) \vee Q(z))$ - *modus ponens* is applied

$f_4 : (\forall z)(P(x) \vee Q(z)) \rightarrow P(x) \vee Q(x)$ - A_4 axiom where x is the term used for the *instantiation of the universal* quantified variable z

$f_3, f_4 \vdash_{mp} f_5 = P(x) \vee Q(x)$ - *modus ponens* is applied

$f_5 \vdash_{univ\_gen} f_6 = (\forall x)(P(x) \vee Q(x))$ - *universal generalization* is applied

$(f_1, f_2, f_3, f_4, f_5, f_6)$ is the deduction (the proof) of $(\forall x)(P(x) \vee Q(x))$ from the hypothesis $(\forall y)(\forall z)(P(y) \vee Q(z))$.

Example 2.4.

Prove that the formula $(\forall x)(P(x) \vee Q(x))$ is derivable from the formula $(\forall y)(\forall z)(P(y) \vee Q(z))$, using the definition of deduction.

We have to prove that: $(\forall y)(\forall z)(P(y) \vee Q(z)) \vdash (\forall x)(P(x) \vee Q(x))$

The sequence $(f_1, f_2, f_3, f_4, f_5, f_6)$ of predicate formulas is build as follows:

$f_1 : (\forall y)(\forall z)(P(y) \vee Q(z))$ – hypothesis

$f_2 : (\forall y)(\forall z)(P(y) \vee Q(z)) \rightarrow (\forall z)(P(x) \vee Q(z))$ - A_4 axiom where x is the term used for the *instantiation of the universal* quantified variable y

$f_1, f_2 \vdash_{mp} f_3 = (\forall z)(P(x) \vee Q(z))$ - *modus ponens* is applied

$f_4 : (\forall z)(P(x) \vee Q(z)) \rightarrow P(x) \vee Q(x)$ - A_4 axiom where x is the term used for the *instantiation of the universal* quantified variable z

$f_3, f_4 \vdash_{mp} f_5 = P(x) \vee Q(x)$ *modus ponens* is applied

$f_5 \vdash_{univ\_gen} f_6 = (\forall x)(P(x) \vee Q(x))$ *universal generalization* is applied

$(f_1, f_2, f_3, f_4, f_5, f_6)$ is the deduction (the proof) of $(\forall x)(P(x) \vee Q(x))$ from $(\forall y)(\forall z)(P(y) \vee Q(z))$.

Theorem 2.1. Theorem of deduction [20]

Let X be a set of predicate formulas and U, V predicate formulas.

If $X \cup \{U\} \vdash V$ then $X \vdash U \rightarrow V$.

Theorem 2.2. Refutation theorem [20]

Let X be a set of predicate formulas and U a predicate formula.

If $X \cup \{\neg U\}$ is inconsistent then $X \vdash U$.

The last theorem is used in proof methods such as: *resolution* and *semantic tableaux method*, called ***refutation proof methods*** and they model 'reductio ad absurdum' (proof by contradiction).

Example 2.5.
Using the theorem of deduction, prove that the formula
$(\forall x)(A(x) \to B(x)) \to ((\forall x)A(x) \to (\forall x)B(x))$ is a theorem.

Step1:
We apply the reverse of the theorem of deduction:
If $\vdash (\forall x)(A(x) \to B(x)) \to ((\forall x)A(x) \to (\forall x)B(x))$ then
$(\forall x)(A(x) \to B(x)) \vdash ((\forall x)A(x) \to (\forall x)B(x))$ then
$(\forall x)(A(x) \to B(x)), (\forall x)A(x) \vdash (\forall x)B(x)$

Step2:
We prove that $(\forall x)(A(x) \to B(x)), (\forall x)A(x) \vdash (\forall x)B(x)$ building the sequence $(f_1, f_2, ..., f_8)$ of predicate formulas:

$f_1 : (\forall x)(A(x) \to B(x))$ - hypothesis

$f_2 : (\forall x)(A(x) \to B(x)) \to (A(y) \to B(y))$ - axiom A_4, x is instantiated with the term y

$f_1, f_2 \vdash_{mp} f_3 = A(y) \to B(y)$

$f_4 : (\forall x)A(x)$ - hypothesis

$f_5 : (\forall x)A(x) \to A(y)$ - axiom A_4, x is instantiated with the term y

$f_4, f_5 \vdash_{mp} f_6 = A(y)$

$f_3, f_6 \vdash_{mp} f_7 = B(y)$

$f_7 \vdash_{univ\_gen} f_8 = (\forall x)\, B(x)$

$(f_1, f_2, ..., f_8)$ is the deduction of $(\forall x)\, B(x)$ from the hypothesis $(\forall x)(A(x) \to B(x))$ and $(\forall x)A(x)$.

Step3:
Using the deduction proved in Step2 and applying twice the theorem of deduction we obtain:

If $(\forall x)(A(x) \to B(x)), (\forall x)A(x) \vdash (\forall x)B(x)$ then
$(\forall x)(A(x) \to B(x)) \vdash (\forall x)A(x) \to (\forall x)B(x)$ then
$\vdash (\forall x)(A(x) \to B(x)) \to ((\forall x)A(x) \to (\forall x)B(x))$

Thus we have proved that the initial formula is a theorem.

2.2. Transformation of natural language sentences into predicate formulas

In the following examples, sentences from natural language are translated into first-order language. We explain how the quantifiers, the variables, the constants, the function symbols and the predicate symbols are used in predicate formulas in order to introduce objects and to express properties and relations among objects.

1. All Computer Science (CS) students are smart.
   $(\forall x)(CS\_student(x) \rightarrow smart(x))$
   - $CS\_student$ and $smart$ are unary predicate symbols, expressing properties of the person x.
2. There is someone who studies at Babes-Bolyai University (BBU) and is smart.
   $(\exists x)(student\_BBU(x) \wedge smart(x))$
   - $student\_BBU$ and $smart$ are unary predicate symbols, expressing properties of the person x.
3. Every child loves anyone who gives the child any present.
   $(\forall x)(\forall y)(\forall z)(child(x) \wedge present(y) \wedge gives(z,x,y) \rightarrow loves(x,z))$
   - $child$, $present$ are unary predicates symbols;
   - $loves$ is a binary predicate symbol ($loves(x,y)$ is true if x *loves* y);
   - $gives$ is a ternary predicate symbol ($gives(z,x,y)$ is true if z *gives* to x the *present* y).
4. John's relatives, except Mary, live in Cluj, some of them like opera music, but all of them like to dance.
   $(\forall x)(relative(John,x) \rightarrow likes\_to\_dance(x) \wedge (\neg equal(x,Mary) \rightarrow lives(x,Cluj))) \wedge (\exists y)(relative(John,y) \wedge likes\_opera(y))$
   - $John, Mary, Cluj$ are constants;
   - $likes\_to\_dance, likes\_opera$ are unary predicate symbols;
   - $relative, lives, equal$ are binary predicate symbols;
   - $relative$ is a symmetric binary relation, thus in a reasoning process involving this relation we have to add the formula: $(\forall x)(\forall y)(relative(x,y) \rightarrow relative(y,x))$;
   - the predicate $equal$ is defined by the following axioms:
     $(\forall x)equal(x,x)$ – reflexivity;
     $(\forall x)(\forall y)(equal(x,y) \rightarrow equal(y,x))$ – symmetry;
     $(\forall x)(\forall y)(\forall z)(equal(x,y) \wedge equal(y,z) \rightarrow equal(x,z))$ – transitivity.

5. If x and y are nonnegative integers and x is greater than y, then x^2 is greater than y^2.

   $(\forall x)(\forall y)(nonneg(x) \wedge nonneg(y) \wedge greater(x,y) \rightarrow greater(square(x), square(y))$

   - function symbol: $square \in F_1$, $square(x) = x^2$;
   - predicate symbols: $nonneg \in P_1$, $nonneg(x)$:"$x > 0$" and $greater \in P_2$, $greater(x,y)$:"$x > y$".

6. In a plane if a line x is parallel to a constant line d then all the lines perpendicular to x are also perpendicular to d.

   $(\forall x)(parallel(x,d) \rightarrow (\forall y)(perpendicular(y,x) \rightarrow perpendicular(y,d)))$

   - *perpendicular* and *parallel* are binary predicate symbols corresponding to the geometric relations. In a reasoning process involving these relations we have to add the formulas expressing the properties of reflexivity (for *parallel*), symmetry (for *parallel* and *perpendicular*), transitivity (for *parallel*).

7. The axioms which define the natural numbers:

   a_1. Every natural number has a unique immediate successor.

   existence: $(\forall x)(\exists y)equal(y, successor(x))$

   uniqueness:

   $(\forall x)(\forall y)(\forall z)(equal(y, successor(x)) \wedge equal(z, successor(x)) \rightarrow equal(y,z))$

   a_2. The number 0 is not the immediate successor of a natural number.

   $\neg(\exists x)egal(0, succesor(x))$, 0 is a constant.

   a_3. Every natural number, except 0, has a unique immediate successor.

   existence: $(\forall x)(\exists y)(\neg equal(0,x) \wedge equal(y, predecessor(x)))$

   uniqueness:

   $(\forall x)(\forall y)(\forall z)(equal(y, predecessor(x)) \wedge equal(z, predecessor(x)) \rightarrow equal(y,z))$

   - unary functions: *successor*, *predecessor*;
   - binary predicate: *equal* is reflexive, symmetric and transitive.

   The following formulas express the equality of the successors and the predecessors of two equal numbers:

   $(\forall x)(\forall y)(equal(x,y) \rightarrow equal(successor(x), successor(y)))$

   $(\forall x)(\forall y)(equal(x,y) \rightarrow equal(predecessor(x), predecessor(y)))$

2.3. The semantics of first-order (predicate) logic

The semantics of predicate logic realize the connection between the constant symbols, the function symbols, the predicate symbols and the real constants, functions, predicates from the modeled universe. Also, a meaning for each formula from the language, in terms of the modeled universe, is provided.

Definition 2.4.

An ***interpretation*** for a language L of predicate logic is a pair $I =< D, m >$, where:

1. D is a nonempty set called the domain of interpretation.
2. m is a function that assigns:
   - a fixed value $m(c) \in D$ to the constant c.
   - a function $m(f): D^n \to D$ to each n-ary function symbol f;
   - a predicate $m(P): D^n \to \{T, F\}$ to each n-ary predicate symbol P.

Notations: $I =< D, m >$ be an interpretation.

- $|I| = D$ is the domain of I, Var is the set of variables.
- $I|X|$ is $m(X)$ where X is a predicate symbol or a function symbol.
- $As(I)$ is the set of assignment functions for variables over the domain of interpretation I.
  An assignment function $fa \in As(I)$ is defined as follows: $fa: Var \to |I|$.
- $[fa]_x = \{fa' | fa' \in As(I)$ and $fa'(y) = fa(y)$, for every $y \neq x\}$.

Definition 2.5.

Let U and V be first-order formulas, A a formula free of the variable x, I an interpretation and $fa \in As(I)$ an assignment function. The evaluation function v^I_{fa} is defined inductively as follows:

1. $v^I_{fa}(x) = fa(x), x \in Var$;
2. $v^I_{fa}(c) = I|c|, c \in Const$;
3. $v^I_{fa}(f(t_1, t_2, \ldots, t_n) = I|g|(v^I_{fa}(t_1), v^I_{fa}(t_2), \ldots, v^I_{fa}(t_n)), g \in F_n, n > 0$;
4. $v^I_{fa}(P(t_1, t_2, \ldots, t_n) = I|P|(v^I_{fa}(t_1), v^I_{fa}(t_2), \ldots, v^I_{fa}(t_n)), P \in \mathrm{P}_n, n > 0$
5. $v^I_{fa}(\neg U) = \neg v^I_{fa}(U)$;
6. $v^I_{fa}(U \wedge V) = v^I_{fa}(U) \wedge v^I_{fa}(V)$;
7. $v^I_{fa}(U \vee V) = v^I_{fa}(U) \vee v^I_{fa}(V)$;
8. $v^I_{fa}(U \to V) = v^I_{fa}(U) \to v^I_{fa}(V)$;
9. $v^I_{fa}((\exists x)A(x)) = T$ if and only if $v^I_{fa'}(A(x)) = T$ for a function $fa' \in [fa]_x$
10. $v^I_{fa}((\forall x)A(x)) = T$ if and only if $v^I_{fa'}(A(x)) = T$ for any function $fa' \in [fa]_x$

Definition 2.6.

1. A first-order formula U is ***satisfiable (consistent)*** if there exists an interpretation I and an assignment function $fa \in As(I)$ such that $v^I_{fa}(U) = T$. Otherwise, the formula is called ***unsatisfiable (inconsistent)***.
2. A first-order formula U ***is true under the interpretation*** I if for any assignment function $fa \in As(I)$, $v^I_{fa}(U) = T$, notation: $\models_I U$, and I is called ***model*** of U.
3. A first-order formula U ***is false under the interpretation*** I if for any assignment function $fa \in As(I)$, $v^I_{fa}(U) = F$, and I is called ***anti-model*** of U.
4. A first-order formula U is ***valid (tautology)*** if U is true under all the possible interpretations, notation: $\models U$.
5. The first-order formulas U and V are ***logically equivalent*** if $v^I_{fa}(U) = v^I_{fa}(V)$ for any interpretation I and any assignment function fa, notation: $U \equiv V$.
6. A set of first-order formulas S ***logically implies*** the first-order formula V if all the models of the set S (the models of the conjunction of all formulas from S) are also models of the formula V. We say that V is a ***logical consequence*** of the set S, notation: $S \models V$.
7. A ***set*** of first-order formulas is ***consistent*** if the conjunction of all its formulas has at least one model. A ***set*** of formulas is ***inconsistent*** if the conjunction of all its formulas does not have a model.

Remarks:

1. The evaluation of a closed formula U depends only on the interpretation in which we want to evaluate it, notation: $v^I(U)$.
2. Any first-order (predicate) formula has an infinite number of interpretations.

Example 2.6.

Build a model and an anti-model for the closed predicate formula:

$U = (\forall x)(P(x) \vee Q(x)) \rightarrow (\forall x)P(x) \vee (\forall x)Q(x)$

- Let us consider the interpretation $I_1 = < D_1, m >$, where:

  $D_1 = \mathbf{N}$ (the set of natural numbers)

  $m(P) : \mathbf{N} \rightarrow \{T, F\}, m(P)(x) = "x \vdots 2"$

  $m(Q) : \mathbf{N} \rightarrow \{T, F\}, m(Q)(x) = "x \vdots 3"$.

  $v^{I_1}(U) = v^{I_1}((\forall x)(P(x) \vee Q(x))) \rightarrow v^{I_1}((\forall x)P(x) \vee (\forall x)Q(x))$

  $= v^{I_1}((\forall x)(P(x) \vee Q(x))) \rightarrow v^{I_1}((\forall x)P(x)) \vee v^{I_1}((\forall x)Q(x))$

  $= (\forall x)_{x \in \mathbf{N}}(x \vdots 2 \ \vee \ x \vdots 3) \rightarrow (\forall x)_{x \in \mathbf{N}}(x \vdots 2) \vee (\forall x)_{x \in \mathbf{N}}(x \vdots 3)$

  $= F \rightarrow F \vee F = F \rightarrow F = T$.

$v^{I_1}(U) = T$, U is evaluated as true under the interpretation I_1 which is a model of U.

- Let us consider the interpretation $I_2 =< D_2, m >$, where:
  $D_2 = \{4, 9\}$ – the domain of interpretation;
  $m(P) : \{4, 9\} \to \{T, F\}, m(P)(x) =" x \vdots 2"$
  $m(Q) : \{4, 9\} \to \{T, F\}, m(Q)(x) =" x \vdots 3"$.
  To evaluate the formula U under the interpretation I_2, with the finite domain $D_2 = \{4, 9\}$, the universally quantified subformulas are replaced by the conjunction of their instances for $x = 4$ and $x = 9$.

$$v^{I_2}(U) = v^{I_2}((\forall x)(P(x) \vee Q(x))) \to v^{I_2}((\forall x)P(x) \vee (\forall x)Q(x))$$
$$= v^{I_2}((\forall x)(P(x) \vee Q(x))) \to v^{I_2}((\forall x)P(x)) \vee v^{I_2}((\forall x)Q(x))$$
$$= (4 \vdots 2 \ \vee \ 4 \vdots 3) \wedge (9 \vdots 2 \ \vee \ 9 \vdots 3) \to (4 \vdots 2 \wedge 9 \vdots 2) \vee (4 \vdots 3 \wedge 9 \vdots 3)$$
$$= (T \vee F) \wedge (F \vee T) \to (T \wedge F) \vee (F \wedge T) = T \wedge T \to F \vee F.$$
$$= T \to F = F$$

I_2 evaluates the formula U as false, I_2 is an anti-model of U and thus U *is not a valid formula, it is a contingent one* (I_1 is a model of U).

Example 2.7.

Evaluate the open formula $U(z)$ under the interpretations I_1 and I_2

$U(z) = (\exists x)(\exists y)P(f(x, y), z)$

1. $I_1 =< D_1, m_1 >$, $D = \mathbf{Z}$ (the set of integer numbers),
   $m_1(f)(x, y) = (x + y)^2$ and $m_1(P)(x, y) :" x = y"$.
   Because $U(z)$ is an open formula, its evaluation depends on the assignment of values (integers) to the free variable z, $fa \in As(I_1)$, where:

$$v^{I_1}_{fa}(U(z)) = (\exists x)_{x \in Z}(\exists y)_{y \in Z} "(x + y)^2 = fa(z)" = \begin{cases} \text{T, if } fa(z) \textit{ is a square} \\ \text{F, otherwise} \end{cases}$$

   $U(z)$ is a consistent formula, but it is not a true formula under the interpretation I_1, therefore I_1 *is not a model of the formula.*

2. $I_2 =< D_2, m_2 >$, $D = \mathbf{Z}$ (the set of integer numbers),
   $m_2(f)(a, b) = a + b$ and $m_2(P)(a, b) :" a = b"$.

$$v^{I_2}_{fa}(U(z)) = (\exists x)_{x \in Z}(\exists y)_{y \in Z} "x + y = fa(z)" = T, \forall fa(z) \in \mathbf{Z}.$$

   The formula $U(z)$ is true under the interpretation I_2: every integer number $fa(z)$ is the sum of two integer numbers x and y.
   $\models_{I_2} U(z)$, therefore I_2 is a model of $U(z)$.

2.4. *Logical equivalences in predicate logic*

- ***Expansion laws***

$(\forall x)A(x) \equiv (\forall x)A(x) \wedge A(t)$ – the universal quantifier is an infinitary conjunction,

$(\exists x)A(x) \equiv (\exists x)A(x) \vee A(t)$ – the existential quantifier is an infinitary disjunction,

where t is a term which does not contain x.

- ***DeMorgan's infinitary laws***

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

- ***Quantifiers interchanging laws***

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

Remark: $(\exists x)(\forall y)B(x,y) \not\equiv (\forall y)(\exists x)B(x,y)$

Quantifiers of the same type commute, but quantifiers of different type do not commute.

- ***The extraction of quantifiers in front of the formula laws***

$A \vee (\exists x)B(x) \equiv (\exists x)(A \vee B(x))$ $\qquad$ $A \vee (\forall x)B(x) \equiv (\forall x)(A \vee B(x))$

$A \wedge (\exists x)B(x) \equiv (\exists x)(A \wedge B(x))$ $\qquad$ $A \wedge (\forall x)B(x) \equiv (\forall x)(A \wedge B(x))$

where A does not contain x as a free variable.

$(\exists x)A(x) \vee B \equiv (\exists x)(A(x) \vee B)$ $\qquad$ $(\forall x)A(x) \vee B \equiv (\forall x)(A(x) \vee B)$

$(\exists x)A(x) \wedge B \equiv (\exists x)(A(x) \wedge B)$ $\qquad$ $(\forall x)A(x) \wedge B \equiv (\forall x)(A(x) \wedge B)$

where B does not contain x as a free variable.

- ***Distributive laws***

$(\exists x)(A(x) \vee B(x)) \equiv (\exists x)A(x) \vee (\exists x)B(x)$ distribution of "$\exists$" over "$\vee$"

$(\forall x)(A(x) \wedge B(x)) \equiv (\forall x)A(x) \wedge (\forall x)B(x)$ distribution of "$\forall$" over "$\wedge$"

The above laws are pairs of dual equivalences. "$\forall$" and "$\exists$" are dual quantifiers.

Remark: The distribution of

"$\exists$" over "$\wedge$", "$\forall$" over "$\vee$", "$\exists$" over "$\rightarrow$", "$\forall$" over "$\rightarrow$"

do not provide valid distributive laws. These combinations of quantifiers and connectives express only semi-distributive laws as follows:

Semi-distributivity of "$\exists$" over "$\wedge$":

$\models (\exists x)(A(x) \wedge B(x)) \rightarrow (\exists x)A(x) \wedge (\exists x)B(x)$ and

$\not\models (\exists x)A(x) \wedge (\exists x)B(x) \rightarrow (\exists x)(A(x) \wedge B(x))$

Semi-distributivity of "$\forall$" over "$\vee$":

$\models (\forall x)A(x) \vee (\forall x)B(x) \rightarrow (\forall x)(A(x) \vee B(x))$ and

$\not\models (\forall x)(A(x) \vee B(x)) \rightarrow (\forall x)A(x) \vee (\forall x)B(x))$

Semi-distributivity of "$\exists$" over "$\rightarrow$":

$\models ((\exists x)A(x) \rightarrow (\exists x)B(x)) \rightarrow (\exists x)(A(x) \rightarrow B(x))$ and

$\not\models (\exists x)(A(x) \rightarrow B(x)) \rightarrow ((\exists x)A(x) \rightarrow (\exists x)B(x))$

Semi-distributivity of "$\forall$" over "$\rightarrow$":

$\models (\forall x)(A(x) \rightarrow B(x)) \rightarrow ((\forall x)A(x) \rightarrow (\forall x)B(x))$ and

$\not\models ((\forall x)A(x) \rightarrow (\forall x)B(x)) \rightarrow (\forall x)(A(x) \rightarrow B(x))$

Example 2.8.

Prove that the universal and existential quantifiers do not commute.

We have to prove that the logical equivalence:

$(\exists x)(\forall y)L(x,y) \equiv (\forall y)(\exists x)L(x,y)$ does not hold.

We choose the interpretation $I =< D, m >$, where:

- D is the set of all persons in the world
- $m(L): D \times D \rightarrow \{T,F\}, m(L)(x,y) =$"$x$ *loves* y"

Under the interpretation I, the formula $U_1 = (\exists x)(\forall y)L(x,y)$ has the meaning: "*There exists a person who loves all persons.*"

The formula $U_2 = (\forall y)(\exists x)L(x,y)$ has the meaning: "*All persons are loved by at least one person.*", under the same interpretation *I.*

These two natural language statements are not equivalent, so $U_1 \not\equiv U_2$, but note that $U_1 \models U_2$.

Example 2.9.

Prove that the formula

$U = (\exists x)A(x) \wedge (\exists x)B(x) \rightarrow (\exists x)(A(x) \wedge B(x))$ is not valid.

Let us consider the interpretation $I =< D, m >$, where D is the set of all straight lines belonging to a plan P.

Let $d \in P$ be a constant object (line) from the domain of interpretation.

$m(A): D \rightarrow \{T,F\}, m(A)(x) =$"$x \perp d$";

$m(B): D \rightarrow \{T,F\}, m(B)(x) =$"$x \parallel d$";

$$v^I(U) = v^I((\exists x)A(x) \wedge (\exists x)B(x)) \rightarrow v^I((\exists x)(A(x) \wedge B(x))) =$$

$$= v^I((\exists x)A(x)) \wedge v^I((\exists x)B(x)) \rightarrow v^I((\exists x)(A(x) \wedge B(x))) =$$

$$= (\exists x)_{x \in D}(x \perp d) \wedge (\exists x)_{x \in D}(x \parallel d) \rightarrow (\exists x)_{x \in D}(x \perp d \ \wedge x \parallel d) =$$

$$= T \wedge T \rightarrow F = T \rightarrow F = F.$$

U is evaluated as false under the interpretation I, so I is an anti-model of U.

We conclude that U *is not a valid formula*, but *it is consistent*, having J as a model.

$J =< D_1, m_1 >$, where:

D_1 - the set of all persons living in Cluj-Napoca.

$m_1(A): D_1 \to \{T, F\}, m_1(A)(x)$: "$x$ owns a car";

$m_1(B): D_1 \to \{T, F\}, m_1(B)(x)$: "$x$ has blue eyes";

$$v^J(U) = v^J((\exists x)A(x) \wedge (\exists x)B(x)) \to v^J((\exists x)(A(x) \wedge B(x))) =$$

$$= v^J((\exists x)A(x)) \wedge v^J((\exists x)B(x)) \to v^J((\exists x)(A(x) \wedge B(x))) =$$

$$= (\exists x)_{x \in D_1}\text{"}x \text{ owns a car"} \wedge (\exists x)_{x \in D_1}\text{"}x \text{ has blue eyes"} \to$$

$$\to (\exists x)_{x \in D_1}\text{"}x \text{ owns a car } \textit{and } x \text{ has blue eyes"}$$

$$= T \wedge T \to T = T \to T = T.$$

U is evaluated as true under the interpretation J, J is a model of U.

Theorem 2.3. [58]

Soundness and completeness theorem states the equivalence between "*logical consequence*" and "*syntactic consequence*" concepts.

Let $U_1,...,U_{n-1},U_n,V$ be first-order formulas.

1. completeness: if $U_1,...,U_{n-1},U_n \models V$ then $U_1,...,U_{n-1},U_n \vdash V$.
2. soundness: if $U_1,...,U_{n-1},U_n \vdash V$ then $U_1,...,U_{n-1},U_n \models V$.

A particular case of this theorem is the following result:

"**A formula is a tautology if and only if it is a theorem in first-order logic.**"

Theorem 2.4. (A. Church, 1936) [13]

The problem of the validity of a first-order formula is ***undecidable***, but it is ***semi-decidable***. If a procedure *Proc* is used to check the validity of a first-order formula U we have the following cases:

1. if U is a valid formula, then *Proc* ends with the corresponding answer.
2. if the formula U is not valid, then *Proc* ends with the corresponding answer or *Proc* may never stop.

The above result can be improved for fragments of first-order logic as follows:

Theorem 2.5. [17]

There is a decision procedure to check the validity of the formulas belonging to the following classes of formulas with the prefix of the prenex form:

1. $\forall^*\exists^* : (\forall x_1)...(\forall x_n)(\exists y_1)...(\exists y_m), m, n \geq 0$;
2. $\forall^*\exists\forall^* : (\forall x_1)...(\forall x_n)(\exists y)(\forall z_1)...(\forall z_m), m, n \geq 0$;
3. $\forall^*\exists\exists\forall^* : (\forall x_1)...(\forall x_n)(\exists y_1)(\exists y_2)(\forall z_1)...(\forall z_m), m, n \geq 0$.

The proof methods used in predicate logic:

- semantic tableaux method (semantic and refutation);
- resolution (syntactic and refutation);
- sequent/anti-sequent calculi (syntactic and direct);
- definition of deduction – axiomatic system (syntactic and direct)
- Herbrand-based procedure (syntactic and refutation)

2.5. Normal forms in first-order logic

The normal forms of predicate formulas are used as input data in proof methods such as *resolution* and *Herbrand-based procedure.*

Definition 2.7.

A predicate formula U is in ***prenex normal form*** if it has the form: $(Q_1x_1)...(Q_nx_n)M$, where $Q_i, i=1,\ldots,n$ are quantifiers, and M is quantifier-free. The sequence $(Q_1x_1)...(Q_nx_n)$ is called the ***prefix* of the formula** U, and M is called the ***matrix* of the formula** U. A predicate formula is in ***conjunctive prenex normal form*** if it is in prenex normal form and the matrix is in CNF.

Theorem 2.6. [20]

A predicate formula admits a logical equivalent conjunctive prenex normal form.

The prenex normal form is obtained by applying transformations which preserve the logical equivalence, according to the following algorithm:

Step 1: The connectives "$\rightarrow$" and "$\leftrightarrow$" are replaced using the connectives "$\neg, \wedge, \vee$".

Step 2: The bound variables are renamed such that they will be distinct.

Step 3: Application of infinitary DeMorgan's laws.

Step 4: The extraction of quantifiers in front of the formula laws are applied.

Step 5: The matrix is transformed into CNF using DeMorgan's laws and the distributive laws.

Remarks:

- After the Step 4 we obtain the prenex normal form which is not unique. If the formula obtained after the second step contains n distinct and independent groups of quantifiers, these groups can be extracted in an arbitrary order, therefore there exist $n!$ prenex normal forms, logically equivalent to the initial formula.
- A conjunctive prenex normal form is obtained after Step 5.

Definition 2.8.

Let U be a first-order formula, and $U^P = (Q_1x_1)...(Q_nx_n)M$ be one of its conjunctive prenex normal form.

1. A formula in ***Skolem normal form***, denoted by U^S corresponds to U and it is obtained as follows:
   For each existential quantifier Q_r from the prefix we apply the transformation:
   - If Q_r is the left-most universal quantifier in the prefix, then we introduce a new constant $\boldsymbol{a}$, and we replace in M all the occurrences of x_r by $\boldsymbol{a}$. (Q_rx_r) is deleted from the prefix.
   - If $Q_{s_1},...,Q_{s_m}, 1 \le s_1 < ... < s_m < r$, are all the universal quantifiers at the left side of Q_r, then we introduce a new m-place function symbol, f, and we replace in M all occurrences of x_r by $f(x_{s_1},...,x_{s_m})$. (Q_rx_r) is deleted from the prefix.

   The constants and functions used to replace the existential quantified variables are called ***Skolem constants*** and ***Skolem functions***. The prefix of the formula U^S contains only universal quantifiers, and the matrix is in conjunctive normal form.
2. A formula in ***clausal normal form*** denoted by U^c corresponds to U and it is obtained by deleting the prefix of U^S.

Remarks:

The transformations used in the Skolemization process do not preserve the logical equivalence but preserve the inconsistency according to the following theorem.

Theorem 2.7.

Let $U_1,U_2,...,U_n,V$ be first-order formulas.

1. V is inconsistent if and only if V^P is inconsistent, if and only if V^S is inconsistent, if and only if V^c is inconsistent.
2. The set $\{U_1,U_2,...,U_n\}$ is inconsistent if and only if the set $\{U_1^c,U_2^c,...,U_n^c\}$ is inconsistent.

Example 2.10.

Transform into prenex normal form and Skolem normal form the formula:

$$U = \neg((\forall x)(P(x) \to Q(x)) \to ((\forall x)P(x) \to (\forall x)Q(x)))$$

$$U = \neg((\forall x)(P(x) \overset{1}{\rightarrow} Q(x)) \overset{2}{\rightarrow} ((\forall x)P(x) \overset{3}{\rightarrow} (\forall x)Q(x)))$$

Step 1: replace the innermost "$\rightarrow$" connectives, denoted by 1 and 3

$$U \equiv \neg((\forall x)(\neg P(x) \vee Q(x)) \rightarrow (\neg(\forall x)P(x) \vee (\forall x)Q(x)))$$

Step 1: replace "$\rightarrow$" connective

$$U \equiv \neg(\neg((\forall x)(\neg P(x) \vee Q(x))) \vee (\neg(\forall x)P(x) \vee (\forall x)Q(x))$$

Step 2: rename the bound variables such that they will be distinct

$$U \equiv \neg(\neg((\forall x)(\neg P(x) \vee Q(x))) \vee (\neg(\forall y)P(y) \vee (\forall z)Q(z))$$

Step 3: apply DeMorgan's laws

$$U \equiv \neg(\neg((\forall x)(\neg P(x) \vee Q(x))) \vee (\neg(\forall y)P(y) \vee (\forall z)Q(z))$$

Step 3: apply DeMorgan's laws

$$U \equiv (\forall x)(\neg P(x) \vee Q(x)) \wedge \neg(\neg(\forall y)P(y) \vee (\forall z)Q(z))$$

Step 4: extract the quantifiers in front of the formula

$$U \equiv (\exists z)(\forall x)(\forall y)((\neg P(x) \vee Q(x)) \wedge P(y) \wedge \neg Q(z)) = U_1 = U_1{}^P$$

$$U \equiv (\forall x)(\exists z)(\forall y)((\neg P(x) \vee Q(x)) \wedge P(y) \wedge \neg Q(z)) = U_2 = U_2{}^P$$

$$U \equiv (\forall x)(\forall y)(\exists z)((\neg P(x) \vee Q(x)) \wedge P(y) \wedge \neg Q(z)) = U_3 = U_3{}^P$$

U_1, U_2, U_3 are three prenex forms of the initial formula U.

Because the formula U contains 3 distinct and independent bound variables, there exists $3! = 6$ prenex normal forms logically equivalent to U.
Note that the matrix is in CNF, and we will not apply Step **5**.

$U_1{}^P, U_2{}^P, U_3{}^P$ are prenex normal forms.

After the Skolemization process, applied to the formulas $U_1{}^P, U_2{}^P, U_3{}^P$ we obtain:

$$U_1{}^S = (\forall x)(\forall y)((\neg P(x) \vee Q(x)) \wedge P(y) \wedge \neg Q(a)), \quad [z \leftarrow a],$$

a is Skolem constant

$$U_2{}^S = (\forall x)(\forall y)((\neg P(x) \vee Q(x)) \wedge P(y) \wedge \neg Q(f(x))), \quad [z \leftarrow f(x)],$$

f is a unary Skolem function

$$U_3{}^S = (\forall x)(\forall y)((\neg P(x) \vee Q(x)) \wedge P(y) \wedge \neg Q(g(x,y))), \quad [z \leftarrow g(x,y)],$$

g is a binary Skolem function

The clausal normal forms are obtained by eliminating the universal quantifiers.

$$U_1{}^C = (\neg P(x) \vee Q(x)) \wedge P(y) \wedge \neg Q(a)$$

$$U_2{}^C = (\neg P(x) \vee Q(x)) \wedge P(y) \wedge \neg Q(f(x))$$

$$U_3{}^C = (\neg P(x) \vee Q(x)) \wedge P(y) \wedge \neg Q(g(x,y))$$

Example 2.11.
Transform into prenex normal form and Skolem normal form the formula:

$U = (\exists x)(\forall y)P(x,y) \vee (\exists z)(\neg Q(z) \vee (\forall u)(\exists t)R(z,u,t))$

$U^P = (\exists x)(\forall y)(\exists z)(\forall u)(\exists t)(P(x,y) \vee \neg Q(z) \vee R(z,u,t))$

$U^S = (\forall y)(\forall u)(P(a,y) \vee \neg Q(f(y)) \vee R(f(y),u,g(y,u)))$, where:

$[x \leftarrow a],[z \leftarrow f(y)],[t \leftarrow g(y,u)]$, a - Skolem constant,

f, g - Skolem functions

$U^C = P(a,y) \vee \neg Q(f(y)) \vee R(f(y),u,g(y,u))$

2.6. Substitutions and unification

In this section we introduce *substitutions* and *unification,* used in predicate resolution (see chapter 5).

Definition 2.9.
A ***substitution*** is a mapping from the set of variables ***Var*** into the set of terms: ***TERMS***. We denote by $\theta = [x_1 \leftarrow t_1, \ldots, x_k \leftarrow t_k]$, a substitution, representing a finite set of replacements of variables, where $x_1, \ldots, x_k$ are distinct variables, $t_1, \ldots, t_k$ are terms, such that $\forall i = 1, \ldots, k : t_i \neq x_i$ and x_i is not a subterm of t_i. $dom(\theta) = \{x_1, \ldots, x_k\}$ is called the ***domain*** of θ.

We use Greek letters: $\varphi, \delta, \phi, \eta, \theta, \lambda$ to stand for substitutions. The empty substitution is denoted by ε.

Definition 2.10.
The result of applying the substitution $\theta = [x_1 \leftarrow t_1, \ldots, x_k \leftarrow t_k]$ to a formula is defined recursively as follows:

1. $\theta(x_i) = t_i, x_i \in dom(\theta)$;
2. $\theta(x) = x, x \notin dom(\theta)$;
3. $\theta(c) = c, c - \text{constant}$;
4. $\theta(f(t_1,\ldots,t_n)) = f(\theta(t_1),\ldots,\theta(t_n)), f \in \mathrm{F}_n$;
5. $\theta(p(t_1,\ldots,t_n)) = p(\theta(t_1),\ldots,\theta(t_n)), p \in \mathrm{P}_n$;
6. $\theta(\neg U) = \neg\theta(U)$;
7. $\theta(U \circ V) = \theta(U) \circ \theta(V), \circ \in \{\vee, \wedge, \rightarrow, \leftrightarrow\}$

Note: An instance $\theta(U)$ of U is obtained by replacing simultaneously each occurrence of x_i in U by t_i.

Definition 2.11.

The ***composition*** of two substitutions $\theta_1 = [x_1 \leftarrow t_1, \ldots, x_k \leftarrow t_k]$ and $\theta_2 = [y_1 \leftarrow s_1, \ldots, y_n \leftarrow s_n]$ is defined as:

$\theta = \theta_1\theta_2 = [x_i \leftarrow \theta_2(t_i) \mid x_i \in dom(\theta_1), x_i \neq \theta_2(t_i)] \cup [y_j \leftarrow s_j \mid y_j \in dom(\theta_2) \setminus dom(\theta_1)]$

Properties: Let $\theta, \theta_1, \theta_2, \theta_3$ be substitutions.

- $\varepsilon\theta = \theta\varepsilon = \theta$; ε - empty substitution;
- $\theta_1(\theta_2\theta_3) = (\theta_1\theta_2)\theta_3$ associativity property;
- $\theta_1\theta_2 \neq \theta_2\theta_1$, the composition of two substitutions is not commutative

Example 2.12.

We prove that the composition of two substitutions is not commutative

$\theta_1 = [x \leftarrow f(y), y \leftarrow f(a), z \leftarrow u]$,

$\theta_2 = [y \leftarrow g(a), u \leftarrow z, v \leftarrow f(f(a))]$

$\lambda_1 = \theta_1\theta_2 = [x \leftarrow \theta_2(f(y)), y \leftarrow \theta_2(f(a)), z \leftarrow \theta_2(u)] \cup [u \leftarrow z, v \leftarrow f(f(a))] =$

$= [x \leftarrow f(g(a)), y \leftarrow f(a), z \leftarrow z, u \leftarrow z, v \leftarrow f(f(a))] =$

$= [x \leftarrow f(g(a)), y \leftarrow f(a), u \leftarrow z, v \leftarrow f(f(a))]$

$\lambda_2 = \theta_2\theta_1 = [y \leftarrow g(a), v \leftarrow f(f(a)), x \leftarrow f(y), z \leftarrow u]$

$\lambda_1 \neq \lambda_2$, so the composition of two substitutions is not commutative in general.

We compute for $U = p(u, v, x, y, z)$ the instances $\lambda_1(U)$ and $\lambda_2(U)$:

$\lambda_1(U) = p(z, f(f(a)), f(g(a)), f(a), z)$

$\lambda_2(U) = p(u, f(f(a)), f(y), g(a), u)$

Definition 2.12.

1. A substitution θ is a ***unifier*** of the terms t_1 and t_2 if $\theta(t_1) = \theta(t_2)$. The term $\theta(t_1)$ is called the ***common instance*** of the unified terms.
2. A ***unifier of a set*** $\{U_1, \ldots, U_n\}$ of formulas is a substitution θ such that $\theta(U_1) = \ldots = \theta(U_n)$.

Example 2.13.

Let x, y be variables, a, b constants, f, g function symbols and P, Q predicate symbols

- the terms $f(x,x)$ and $f(a,b)$ cannot be unified because they do not have a common instance if a, b are distinct constants.
- $P(x)$ and $P(f(x))$ are not unifiable because x is a subterm of $f(x)$.
- the atoms $P(x)$ and $Q(a)$ are not unifiable because they do not have the same predicate symbol.
- the atoms $P(x, y, z)$ and $P(a, b)$ are not unifiable because the predicate symbols have different arity.

- $t_1 = f(x,b)$, $t_2 = f(a,y)$ have a common instance $f(a,b) = \theta(t_1) = \theta(t_2)$, where $\theta = [x \leftarrow a, y \leftarrow b]$ is a unifier for these two terms.
- $t_1 = g(g(x))$ and $t_2 = g(y)$ have many unifiers:
  $\mu_1 = [y \leftarrow g(x)]$, the common instance: $g(g(x))$
  $\mu_2 = [x \leftarrow 5, y \leftarrow g(5)]$, the common instance: $g(g(5))$.
  μ_1 is more general than μ_2.

Definition 2.13.
A ***most general unifier*** (*mgu*) is a unifier μ such that any other unifier θ can be obtained from μ by a further substitution λ, $\theta = \mu\lambda$.

Algorithm for computing the mgu of two literals [62]:
input: $l_1 = P_1(t_{1_1}, t_{1_2}, \ldots, t_{1_n})$ and $l_2 = P_2(t_{2_1}, t_{2_2}, \ldots, t_{2_k})$ two literals
output: $mgu(l_1, l_2)$ or the message "l_1, l_2 are not unifiable"
begin
  if ($P_1 \neq P_2$) // the predicate symbols are different
    then write "l_1, l_2 are not unifiable"; exit;
  end_if
  if ($n \neq k$)
    then write "l_1, l_2 are not unifiable"; exit;
  end_if
  $\theta := \varepsilon$; // initialization with empty substitution
  while ($\theta(l_1) \neq \theta(l_2)$)
    find the terms corresponding to the outermost function symbols or variables that are different in $\theta(l_1)$, $\theta(l_2)$ and denote them by t_1 and t_2.
    if (neither one of t_1 and t_2 is a variable or one is a subterm of the other one)
      then write "l_1 and l_2 are not unifiable"; exit;
    end_if
    if (t_1 is a variable) // λ is the unifier of the terms t_1, t_2 in the current iteration
      then $\lambda := [t_1 \leftarrow t_2]$;
      else $\lambda := [t_2 \leftarrow t_1]$;
    end_if
    $\theta := \theta\lambda$;
    if (θ is not a substitution)
      then write "l_1 and l_2 are not unifiable"; exit;
    end_if
  end_while
  write "l_1 and l_2 are unifiable and θ is $mgu(l_1, l_2)$"
end

Remark:
The most general unifier of two literals may be not unique: if during an iteration, the terms that must be unified t_1 and t_2 are both variables we can replace t_1 by t_2 or t_2 by t_1.

Example 2.14.
Find the most general unifier of the literals l_1 and l_2.

$l_1 = P(a, x, f(g(y)))$, $\quad l_2 = P(y, f(z), f(z))$,

where $x, y, z \in Var$, $a \in Const$, $f, g \in \mathrm{F}_1$, $P \in \mathrm{P}_3$.

The algorithm presented before is applied. In each iteration, the terms that are unified are underlined.

$\theta := \varepsilon$

$\theta(l_1) = P(\underline{a}, x, f(g(y)))$ and $\theta(l_2) = P(\underline{y}, f(z), f(z))$,

first iteration:

$\lambda := [y \leftarrow a]$;

$\theta := \theta\lambda = [y \leftarrow a]$;

$\theta(l_1) = P(a, \underline{x}, f(g(a)))$ and $\theta(l_2) = P(a, \underline{f(z)}, f(z))$

second iteration:

$\lambda := [x \leftarrow f(z)]$;

$\theta := \theta\lambda = [y \leftarrow a][x \leftarrow f(z)] = [y \leftarrow a, x \leftarrow f(z)]$;

$\theta(l_1) = P(a, f(z), f(\underline{g(a)}))$ and $\theta(l_2) = P(a, f(z), f(\underline{z}))$

third iteration:

$\lambda := [z \leftarrow g(a)]$

$\theta := \theta\lambda =$

$= [y \leftarrow a, x \leftarrow f(z)][z \leftarrow g(a)] = [y \leftarrow a, x \leftarrow f(g(a)), z \leftarrow g(a)] = mgu(l_1, l_2)$

l_1 and l_2 are unifiable and their common instance is:

$\theta(l_1) = \theta(l_2) = p(a, f(g(a)), f(g(a)))$.

Example 2.15.
Check if the literals l_1 and l_2 are unifiable:

$l_1 = Q(x, a, f(x, y))$ and $l_2 = Q(b, y, f(z, c))$,

where $x, y, z \in Var$, $a, b, c \in Const$, $f \in \mathrm{F}_2$, $Q \in \mathrm{P}_3$.

The algorithm for computing the *mgu* of two literals is applied:

$\theta := \varepsilon$;

$\theta(l_1) = Q(\underline{x}, a, f(x, y))$ and $\theta(l_2) = Q(\underline{b}, y, f(z, c))$;

first iteration:

$\lambda := [x \leftarrow b]$, unifier of the terms x (variable) and b (constant);

$\theta := \theta\lambda = [x \leftarrow b]$;

$\theta(l_1) = Q(b, \underline{a}, f(b,y))$ and $\theta(l_2) = Q(b, \underline{y}, f(z,c))$;

second iteration:

$\lambda = [y \leftarrow a]$, unifier of the terms y (variable) and a (constant)

$\theta := \theta\lambda$

$\theta := [x \leftarrow b][y \leftarrow a] = [x \leftarrow b, y \leftarrow a]$,

$\theta(l_1) = Q(b, a, f(\underline{b}, a))$ and $\theta(l_2) = Q(b, a, f(\underline{z}, c))$;

third iteration:

$\lambda = [z \leftarrow b]$, unifier of the terms z (variable) and b (constant)

$\theta := \theta\lambda$

$\theta := [x \leftarrow b, y \leftarrow a][z \leftarrow b] = [x \leftarrow b, y \leftarrow a, z \leftarrow b]$

$\theta(l_1) = Q(b, a, f(b, \underline{a}))$ and $\theta(l_2) = Q(b, a, f(b, \underline{c}))$;

fourth iteration:

The terms a and c, which are distinct constants, are not unifiable, so the terms $f(b,a)$ and $f(z,c)$ corresponding to the third argument of the literals are not unifiable.

The conclusion is that the *literals l_1 and l_2 are not unifiable.*

2.7. Herbrand – based procedure

Herbrand-based procedure [12] is a refutation proof method used to solve the decision problem in first-order logic.

Definition 2.14.

1. A ***ground term*** or a ***ground atom*** are free of variables.
2. A formula is called ***ground formula*** if it does not contain any variable or quantifier.
3. A^g is a ***ground instance*** of the quantifier-free formula A, if A^g is obtained by replacing all free variables from A by ground terms.

Example 2.16.

- $a, f(a), g(f(a),b)$ – ground terms, where:
  - a, b - constants, f, g - function symbols
- $P(a), Q(f(a),b)$ – ground atoms, where:
  - a, b - constants, f - function symbol, P, Q - predicate symbols
- $A^g = P(a) \wedge \neg Q(b, f(a))$ is a ground instance of the formula $A(x,y,z) = P(x) \wedge \neg Q(y, f(z))$, $[x \leftarrow a, y \leftarrow b, z \leftarrow a]$.

Definition 2.15. [44]

Let $S = \{A_1, A_2, \ldots, A_n\}$ be a set of first-order quantifier-free formulas.

1. The ***Herbrand universe of*** S, denoted by H_S, is the set of all ground terms built using the constants and the function symbols from S as follows:
   $H_S = \bigcup_{i \geq 0} H_i$, where:
   $H_0 = C =$ the set of all constants of S;
   if $C = \varnothing$ then $H_0 = \{a\}$, a is a new constant;
   $H_{i+1} = H_i \cup \{f(t_1,\ldots,t_k) \mid t_1,\ldots,t_k \in H_i, f \in F_k,\ f - \textit{function symbol of } S\}$.
2. The ***Herbrand base*** of S, denoted by BH_S, is the set of all ground atoms built using the predicate symbols of S and the ground terms of H_S.
   $BH_S = \{P(t_1,\ldots,t_k) \mid t_1,\ldots,t_k \in \mathrm{H_S}, P \in \mathrm{P}_k,\ P - \textit{predicate symbol of } S\}$.
3. The ***Herbrand system*** of S, denoted by SH_S, is the set of all ground instances of the formulas from S, obtained by replacing the free variables by ground terms of H_S.

Example 2.17.

1. $S_1 = \{P(a) \vee \neg Q(x), R(y)\}$
   $H_{S_1} = \{a\}$ - the Herbrand universe is finite.
   $BH_{S_1} = \{P(a), Q(a), R(a)\}$ - the Herbrand base
   $SH_{S_1} = \{P(a) \vee \neg Q(a), R(a)\}$ - the Herbrand system
2. $S_2 = \{P(f(x)), \neg Q(z, y)\}$
   $H_{S_2} = \{c, f(c), f(f(c)), \ldots\}$; c - constant, the Herbrand universe is infinite
   $BH_{S_2} = \{P(f(c)), Q(c,c), P(f(f(c)), Q(c, f(c)), Q(f(c), c), Q(f(c), f(c)), \ldots\}$
   - the Herbrand base is infinite
   $SH_{S_2} = \{P(f(c)), \neg Q(c,c), P(f(f(c)), \neg Q(c, f(c)), \neg Q(f(c), c), \neg Q(f(c), f(c)), \ldots\}$
   - the Herbrand system is infinite

Theorem 2.8. Herbrand's theorem[12]

Let S be a set of first-order quantifier-free formulas. S admits a model if and only if the set SH_S (the Herbrand system of S) admits a model.

Theorem 2.9. [12]

A set S of first-order quantifier-free formulas is inconsistent *if and only if* the set SH_S (the Herbrand system of S) is inconsistent *if and only if* there is a finite inconsistent subset of SH_S.

Remark:
The problem of checking the inconsistency of a set of first-order formulas was reduced to checking the inconsistency of a set (finite or infinite) of propositional formulas according to:

S inconsistent if and only if the Herbrand system SH_{S_1} is inconsistent,

where $S_1 = S^c$, S^c is the set of clausal normal forms of the formulas from S.

Herbrand's theorem suggests a refutation proof procedure based on the following theoretical result:

$U_1, U_2, ..., U_n \vdash V$ *if and only if*

$S = \{U_1^c, U_2^c, ..., U_n^c, (\neg V)^c\}$ is inconsistent *if and only if*

SH_S (Herbrand system of S) is inconsistent *if and only if*

there is a finite inconsistent subset of SH_S.

Herbrand-based algorithm:

input: $U_1, U_2, ..., U_n, V$ - first-order formulas

output: the message $U_1, U_2, ..., U_n \vdash V$ or $U_1, U_2, ..., U_n \nvdash V$

begin

build $S = \{U_1^c, U_2^c, ..., U_n^c, (\neg V)^c\}$

build SH_S;

if SH_S is finite

then

if (SH_S is inconsistent)

then write "$U_1, U_2, ..., U_n \vdash V$ "; exit;

else write "$U_1, U_2, ..., U_n \nvdash V$ "; exit;

end_if

else // $SH_S = \{Z_1, Z_2, \ldots, Z_m, \ldots\}$ is infinite

$k := 1$;

while ($Z_1 \wedge Z_2 \wedge \ldots \wedge Z_k$ is consistent) (**)

$k := k + 1$;

end_while

write "$U_1, U_2, ..., U_n \nvdash V$ "; exit;

end_if

end

This is a semi-decision procedure which never stops if the set SH_S is an infinite consistent set (**). For implementation, the loop "while" must contain a restriction for the iterations number, to avoid the infinite loop. If we exit the loop because the maximum number of iterations was reached, we cannot decide if SH_S is inconsistent or not, and thus we cannot decide if $U_1,U_2,...,U_n \vdash V$ or $U_1,U_2,...,U_n \nvdash V$.

Example 2.18.

Using the Herbrand-based algorithm check the validity of the predicate formula:

$U = (\forall x)(P(x) \to Q(x)) \to ((\forall x)P(x) \to (\forall x)Q(x))$.

Let us denote $A = \neg U$ and we transform A into clausal normal form.

$$A = \neg U \equiv \neg((\forall x)(P(x) \to Q(x)) \to ((\forall x)P(x) \to (\forall x)Q(x))) \equiv$$
$$\equiv (\forall x)(\neg P(x) \vee Q(x)) \wedge \neg((\forall x)P(x) \to (\forall x)Q(x)) \equiv$$
$$\equiv (\forall x)(\neg P(x) \vee Q(x)) \wedge (\forall x)P(x) \wedge \neg(\forall x)Q(x)) \equiv$$
$$\equiv (\forall x)(\neg P(x) \vee Q(x)) \wedge (\forall x)P(x) \wedge (\exists x)\neg Q(x) \equiv$$
$$\equiv (\forall x)(\neg P(x) \vee Q(x)) \wedge (\forall y)P(y) \wedge (\exists z)\neg Q(z) \equiv$$

We consider two prenex normal forms:

$$A_1^P = (\exists z)(\forall x)(\forall y)((\neg P(x) \vee Q(x)) \wedge P(y) \wedge \neg Q(a))$$

$$A_2^P = (\forall x)(\exists z)(\forall y)((\neg P(x) \vee Q(x)) \wedge P(y) \wedge \neg Q(a))$$

The corresponding Skolem and clausal forms are as follows:

$A_1^S = (\forall x)(\forall y)((\neg P(x) \vee Q(x)) \wedge P(y) \wedge \neg Q(a))$, a – Skolem constant

$$A_1^C = (\neg P(x) \vee Q(x)) \wedge P(y) \wedge \neg Q(a)$$

$A_2^S = (\forall x)(\forall y)((\neg P(x) \vee Q(x)) \wedge P(y) \wedge \neg Q(f(x)))$, f – unary Skolem function

$$A_2^C = (\neg P(x) \vee Q(x)) \wedge P(y) \wedge \neg Q(f(x))$$

We consider the following two sets of clauses:

$S_1 = \{\neg P(x) \vee Q(x), P(y), \neg Q(a)\}$ and

$S_2 = \{\neg P(x) \vee Q(x), P(y), \neg Q(f(x))\}$

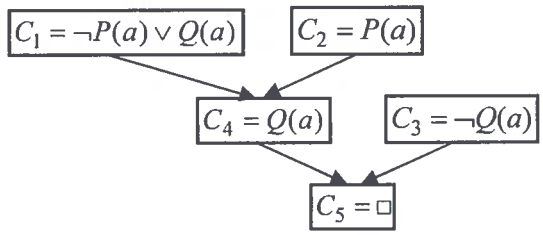
"U is valid if and only if SH_S is inconsistent", where S is S_1 or S_2.

S_1 is used in the Herbrand-based algorithm:

$H_{S_1} = \{a\}$, $SH_{S_1} = \{C_1 = \neg P(a) \vee Q(a), C_2 = P(a), C_3 = \neg Q(a)\}$

The Herbrand universe and Herbrand system are finite.

We derive the empty clause using propositional resolution (see Chapter 5) as follows:



SH_{S_1} is an inconsistent set, therefore A is inconsistent and U is a valid formula.

S_2 is used in the Herbrand-based algorithm:

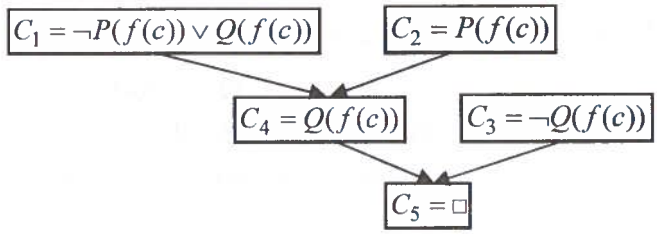
$$H_{S_2} = \{c, f(c), f(f(c)), ...\}$$

$$SH_{S_2} = \{\neg P(c) \vee Q(c), P(c), \underline{\neg Q(f(c))}, \underline{\neg P(f(c)) \vee Q(f(c))}, \underline{P(f(c))}, \underline{\neg Q(f(f(c)))}, ...\}$$

The Herbrand universe and Herbrand system are infinite.

We consider the finite set of clauses:

$W = \{P(f(c)), \neg Q(f(c)), \neg P(f(c)) \vee Q(f(c))\}$ which is a subset of SH_{S_2}. From W we can derive $\square$ as follows:



Thus, W is an inconsistent set, and SH_{S_2} is inconsistent because it has an inconsistent subset. The conclusion is that A is an inconsistent formula and U is a tautology.

If we want to avoid working with an infinite set of clauses, caused by the existence of function symbols, we must choose the Skolem form containing only Skolem constants.

Example 2.19.

Check the consistency/inconsistency of the following set of first-order quantifier-free formulas:

$$S = \{P(x) \rightarrow Q(x) \vee R(x), Q(y) \rightarrow R(y), R(a), \neg P(a)\}.$$

We transform the formulas into clausal forms:

$$S = \{\neg P(x) \vee Q(x) \vee R(x), \neg Q(y) \vee R(y), R(a), \neg P(a)\}$$

$H_S = \{a\}$ - Herbrand universe

$SH_s = \{C_1 = \neg P(a) \vee Q(a) \vee R(a), C_2 = \neg Q(a) \vee R(a), C_3 = R(a), C_4 = \neg P(a)\}$

- Herbrand system contains propositional clauses.

We try to simplify SH_S (see Theorem 5.6):

$\neg P(a)$ is a pure literal and we eliminate the clauses (C_1 and C_4) containing it.

$R(a)$ is also a pure literal and we eliminate C_2 and C_3.

After applying these transformations on SH_S, we obtain $SH_S' = \varnothing$, and thus SH_S is a consistent set, therefore S is a consistent set of clauses.

2.8. Exercises

Exercise 2.1.

Transform the following sentences from natural language into predicate formulas. Explain the syntactic elements used in the predicate formulas: variables, constants, functions symbols, predicate symbols.

1. In a plane if a line x is perpendicular to a constant line d then all the lines parallel to x are perpendicular to d.
2. In a plane there are lines parallel to a constant line d and there are lines perpendicular to d.
3. If x is an integer divisible by 10, it can be decomposed in two factors such that one is divisible by 2 and the other one is divisible by 5, and x can be written as a sum of 2 even numbers.
4. Every positive number can be written as a product of two positive numbers and as a product of two negative numbers.
5. If x and y are positive prime numbers, $x \neq y$, $x \neq 2$ and $y \neq 2$, their sum and difference are even numbers and their product is an odd number.
6. For every positive integer x, if x is a square of an integer, then there exists an integer y such that $(y+1)*(y-1) = x-1$.
7. For every positive integer x, if x is not a prime, then there exists a prime y such that y divides x and y is smaller than x.
8. The sum of two even numbers is an even number and their product is divisible by 4.

Exercise 2.2.

Transform the following statements from natural language into predicate formulas choosing the appropriate constants, function symbols and predicate symbols:

1. Every student who makes good grades is brilliant or studies.
2. Some of John's colleagues like to draw and some like to dance.
3. CS students like either algebra or logic, all of them like Java but only Bill likes history.

4. All Mary's relatives live in Cluj-Napoca, only her cousin John lives in Bucharest.
5. Anyone who owns a rabbit hates anything that chases any rabbit.
6. All birds have wings but only penguins do not fly.
7. Santa has some reindeer with a red nose, then every child loves Santa.
8. Every investor who bought something that falls is not happy.
9. Anyone who has any cats will not have any mice.
10. Caterpillars and snails are much smaller than birds, which are much smaller than foxes, which in turn are much smaller than wolves.
11. Caterpillars and snails like to eat some plants.
12. Every animal either likes to eat all plants or all animals much smaller than itself that like to eat some plants.

Exercise 2.3.
Using the definition of deduction in predicate logic prove:

1. $(\forall y)(\forall z)(P(y) \wedge Q(z)) \vdash (\forall x)(P(x) \wedge Q(x))$;
2. $(\forall y)(P(y) \vee Q(y)), \neg(\forall z)P(z) \vdash (\exists x)Q(x)$;
3. $(\exists y)P(y) \rightarrow (\forall z)Q(z) \vdash (\forall x)(P(x) \rightarrow Q(x))$;
4. $(\forall y)(P(y) \rightarrow Q(y)), (\forall z)P(z) \vdash (\forall x)Q(x)$;
5. $(\forall y)P(y) \vdash (\forall x)(Q(x) \rightarrow P(x))$;
6. $(\forall y)P(y) \vdash (\forall x)(Q(x) \vee P(x))$;
7. $(\forall y)P(y) \vdash (\exists x)P(x)$;
8. $(\forall y)(\forall z)(P(y) \rightarrow Q(z)) \vdash (\forall x)(P(x) \rightarrow Q(x))$.

Exercise 2.4.
Using the given interpretations evaluate the following formulas:

1. $U = (\exists x)A(x) \wedge (\exists x)B(x) \rightarrow (\forall x)(A(x) \vee B(x))$
   Interpretation $I =< D, m >$, where:
   $D =$ the set of all straight lines of a plan P
   Let $d \in P$, a constant straight line belonging to the interpretation domain
   $m(A): D \rightarrow \{T, F\}, m(A)(x): "x \perp d"$; $m(B): D \rightarrow \{T, F\}, m(B)(x): "x \| d"$;
2. $U = (\exists x)(P(x) \wedge Q(x)) \rightarrow (\exists x)P(x) \vee Q(12)$
   Interpretation $I =< D, m >$ where:
   $D = \mathbf{N}$ (the set of natural numbers)
   $m(P): \mathbf{N} \rightarrow \{T, F\}, m(P)(x): "x : 5"$; $m(Q): \mathbf{N} \rightarrow \{T, F\}, m(Q)(x): "x : 7"$;
3. $U(z) = (\exists x)(\forall y)(\exists z)P(f(x, y), z)$
   Interpretation $I =< D, m >$, where:
   $D = \mathbf{Z}$ (the set of integer numbers),

$m(f):\mathbf{Z}^2\to\mathbf{Z}$, $m(f)(x,y)=(x+y)^2$ and

$m(P):\mathbf{Z}^2\to\{T,F\}, m(P)(x,y):"x>y"$;

4. $U=(\forall x)(\exists y)P(x,y)\to(\exists y)(\forall x)P(x,y)$

   Interpretation $I=<D,m>$, where:

   $D=$ the set of all triangles,

   $m(P):D^2\to\{T,F\}, m(P)(x,y):"Area(x)\le Area(y)"$;

5. $U=(\forall x)(\exists y)(\exists z)P(x,f(g(y),g(z)))$

   Interpretation $I=<D,m>$, where:

   $D=\mathbf{N}$ (the set of natural numbers)

   $m(f):\mathbf{N}^2\to\mathbf{N}$, $m(f)(x,y)=x+y$ and

   $m(g):\mathbf{N}\to\mathbf{N}$, $m(g)(x)=x^2$ and

   $m(P):\mathbf{N}^2\to\{T,F\}, m(P)(x,y):"x=y"$;

6. $U=((\forall x)A(x)\leftrightarrow(\forall x)B(x))\to(\forall x)(A(x)\leftrightarrow B(x))$

   Interpretation $I=<D,m>$, where:

   $D=$ the set of all persons from Cluj county

   $m(A):D\to\{T,F\}, m(A)(x)$: "the person x has a driver license";

   $m(B):D\to\{T,F\}, m(B)(x)$: "the person x owns a car".

7. $U(z)=(\exists x)(\forall y)(\forall z)P(x,f(y,z))$

   Interpretation $I=<D,m>$, where:

   $D=\mathbf{Z}$ (the set of integer numbers),

   $m(f):\mathbf{Z}^2\to\mathbf{Z}$, $m(f)(x,y)=(x+y)^2$ and

   $m(P):\mathbf{Z}^2\to\{T,F\}, m(P)(x,y):"x<y"$;

8. $U=((\forall x)A(x)\to(\exists x)B(x))\to((\exists x)A(x)\to(\forall x)B(x))$

   Interpretation $I=<D,m>$, where:

   $D=$ the set of all persons from Romania

   $m(A):D\to\{T,F\}, m(A)(x)$: "the person x lives in a city";

   $m(B):D\to\{T,F\}, m(B)(x)$: "the person x has a job".

Exercise 2.5.

Prove that the following formulas are not valid by finding anti-models for them.

1. $U_1=((\exists x)P(x)\to(\exists x)Q(x))\to(\forall x)(P(x)\to Q(x))$
2. $U_2=(\exists x)(P(x)\to Q(x))\to((\exists x)P(x)\to(\exists x)Q(x))$;

3. $U_3 = ((\forall x)P(x) \to (\forall x)Q(x)) \to (\forall x)(P(x) \to Q(x))$;
4. $U_4 = (\exists x)P(x) \wedge (\exists x)Q(x) \to (\exists x)(P(x) \wedge Q(x))$;
5. $U_5 = (\forall x)(P(x) \vee Q(x)) \to (\forall x)P(x) \vee (\forall x)Q(x)$;
6. $U_6 = ((\forall x)P(x) \to (\exists x)Q(x)) \to (\forall x)(P(x) \to Q(x))$;
7. $U_7 = ((\exists x)P(x) \to (\exists x)Q(x)) \to (\forall x)(P(x) \to Q(x))$;
8. $U_8 = (\exists x)P(x) \wedge (\exists x)Q(x) \to (\forall x)(P(x) \wedge Q(x))$.

Exercise 2.6.

Choose an arbitrary interpretation for the formula $U_j, j \in \{1,2,...,8\}$ and prove that it is a model of $U_j, j \in \{1,2,...,8\}$.

1. $U_1 = (\forall x)(A(x) \leftrightarrow B(x)) \to ((\forall x)A(x) \leftrightarrow (\forall x)B(x))$;
2. $U_2 = (\forall x)(A(x) \to B(x)) \to ((\forall x)A(x) \to (\forall x)B(x))$;
3. $U_3 = (\forall x)(A(x) \leftrightarrow B(x)) \to ((\exists x)A(x) \leftrightarrow (\exists x)B(x))$;
4. $U_4 = (\exists x)(A(x) \to B(x)) \leftrightarrow ((\forall x)A(x) \to (\exists x)B(x))$;
5. $U_5 = ((\exists x)A(x) \to (\forall x)B(x)) \to (\forall x)(A(x) \to B(x))$;
6. $U_6 = (\forall x)(A(x) \vee B(x)) \to ((\forall x)A(x) \vee (\exists x)B(x))$;
7. $U_7 = (\forall x)(A(x) \to B(x)) \to ((\exists x)A(x) \to (\exists x)B(x))$;
8. $U_8 = (\forall x)(A(x) \to B(x)) \to ((\forall x)A(x) \to (\exists x)B(x))$.

All the formulas from Exercise 2.6 are tautologies.

Exercise 2.7.

Transform the following formulas into prenex, Skolem and clausal normal forms.

1. $U_1 = (\exists x)\,(\neg(\exists y)p(y) \to (\forall y)(q(y) \to r(x)))$;
2. $U_2 = (\exists x)\,((\exists y)p(y) \to \neg(\forall y)(q(y) \to r(x)))$;
3. $U_3 = (\forall x)\,(\neg(\exists y)p(y) \to (\forall y)(q(y) \to r(x)))$;
4. $U_4 = (\forall x)\,((\exists y)p(y) \to \neg(\forall y)(q(y) \to r(x)))$;
5. $U_5 = (\exists x)\,((\forall y)p(y) \to \neg(\exists y)(q(y) \to r(x)))$;
6. $U_6 = (\forall x)\,(\neg(\forall y)p(y) \to (\exists y)(q(y) \to r(x)))$;
7. $U_7 = (\forall x)\,((\forall y)p(y) \to \neg(\exists y)(q(y) \to r(x)))$;
8. $U_8 = (\exists x)\,(\neg(\forall y)p(y) \to (\exists y)(q(y) \to r(x)))$.

Exercise 2.8.

Transform the following formulas into prenex, Skolem and clausal normal forms.

1. $U_1 = (\forall x)(\forall y)((\exists z)p(z) \wedge (\exists u)(q(x,u) \to (\exists z)q(y,z)))$;
2. $U_2 = (\exists x)(\forall y)((\exists z)p(z) \wedge (\exists u)(q(x,u) \to (\exists z)q(y,z)))$;

3. $U_3 = (\forall x)(\exists y)((\exists z)p(z) \wedge (\exists u)(q(x,u) \rightarrow (\exists z)q(y,z)))$;
4. $U_4 = (\exists x)(\exists y)((\exists z)p(z) \wedge (\forall u)(q(x,u) \rightarrow (\exists z)q(y,z)))$;
5. $U_5 = (\forall x)(\exists y)((\exists z)p(z) \wedge (\forall u)(q(x,u) \rightarrow (\exists z)q(y,z)))$;
6. $U_6 = (\forall x)(\forall y)((\exists z)p(z) \wedge (\forall u)(q(x,u) \rightarrow (\exists z)q(y,z)))$;
7. $U_7 = (\forall x)(\forall y)((\exists z)p(z) \wedge (\exists u)(q(x,u) \rightarrow (\forall z)q(y,z)))$;
8. $U_8 = (\exists x)(\forall y)((\exists z)p(z) \wedge (\forall u)(q(x,u) \rightarrow (\exists z)q(y,z)))$.

Exercise 2.9.

Are the literals from the following pairs unifiable? If yes, find their most general unifier. $x, y, z \in Var$, $a, b \in Const$, $f, g \in \mathrm{F}_1$, $h \in \mathrm{F}_2$, $P \in \mathrm{P}_3$.

1. $P(a, x, g(g(y)))$ and $P(y, f(z), f(z))$;
   $P(x, g(f(a)), f(x))$ and $P(f(y), z, y)$;
   $P(a, x, g(g(y)))$ and $P(z, h(z,u), g(u))$;
2. $P(a, x, f(g(y)))$ and $P(y, f(z), f(z))$;
   $P(x, g(f(a)), f(b))$ and $P(f(y), z, z)$;
   $P(a, x, f(g(y)))$ and $P(z, h(z,u), f(b))$;
3. $P(a, f(x), g(h(y)))$ and $P(y, f(z), g(z))$;
   $P(x, g(f(a)), h(x,y))$ and $P(f(z), g(z), y)$;
   $P(g(y), x, f(g(y)))$ and $P(z, h(z,u), f(u))$;
4. $P(a, g(x), f(g(y)))$ and $P(y, z, f(z))$;
   $P(b, g(f(a)), z)$ and $P(f(y), z, g(y))$;
   $P(a, h(x,b), f(g(y)))$ and $P(z, h(z,u), f(u))$;
5. $P(a, x, g(f(y)))$ and $P(f(z), z, g(x))$;
   $P(a, x, g(f(y)))$ and $P(x, y, g(f(b)))$;
   $P(a, h(x,u), g(f(z)))$ and $P(y, h(y, f(z)), g(x))$;
6. $P(a, y, g(f(z)))$ and $P(z, f(z), x)$;
   $P(y, f(x), z)$ and $P(y, f(y), f(y))$;
   $P(h(x,y), x, y)$ and $P(h(y,x), f(z), z)$;
7. $P(a, x, g(f(y)))$ and $P(f(y), z, x)$;
   $P(x, a, g(b))$ and $P(f(y), f(y), g(x))$;
   $P(h(x,a), f(z), z)$ and $P(h(f(y), x), f(x), a)$;

8. $P(a,x,g(f(y)))$ and $P(f(y),f(z),g(z))$;
   $P(x,g(f(a)),x)$ and $P(f(y),z,h(y,f(y)))$;
   $P(a,h(x,u),f(g(y)))$ and $P(z,h(z,u),g(u))$.

Exercise 2.10.

Using the Herbrand-based algorithm check whether the following sets of formulas are inconsistent:

1. $S_1 = \{P(x) \to Q(x) \vee R(x), Q(y) \to R(y), P(a) \wedge \neg R(a)\}$;
2. $S_2 = \{\neg P(a) \to Q(x) \vee R(x), Q(y) \to R(y), \neg P(x) \wedge \neg R(a)\}$;
3. $S_3 = \{P(x) \to Q(x) \wedge R(x), Q(y) \to R(y), P(a) \wedge \neg R(a)\}$;
4. $S_4 = \{P(x) \wedge Q(x) \to R(x), \neg Q(y) \to R(y), P(a) \wedge \neg R(a)\}$;
5. $S_5 = \{P(a) \wedge Q(x) \to R(x), \neg Q(y) \to R(y), P(x) \wedge \neg R(a)\}$;
6. $S_6 = \{P(x) \vee Q(x) \to R(x), \neg Q(y) \to R(y), P(a) \wedge \neg R(a)\}$;
7. $S_7 = \{P(a) \vee Q(x) \to R(x), \neg Q(y) \to R(y), P(x) \wedge \neg R(a)\}$;
8. $S_8 = \{P(x) \wedge \neg Q(x) \to R(x), Q(y) \to R(y), P(a) \wedge \neg R(a)\}$.

3. SEMANTIC TABLEAUX METHOD

The semantic tableaux method is a refutation proof method introduced by Raymond Smullyan [58] for classical logics. It is an efficient method, based on semantic considerations, easily adapted to nonstandard logics (modal, temporal, many-valued, nonmonotonic) [34, 35, 49, 56]. Its basic aim is to decide the consistency (satisfiability) and to find all the models of a formula.

3.1. A Classical Approach to the Semantic Tableaux Method

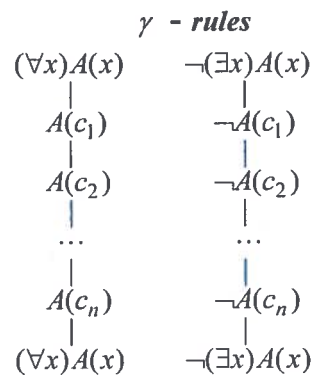
Based on semantic considerations, a predicate formula belongs to one of the following four classes:

- α - ***class*** contains conjunctive formulas:
  $A \wedge B$, $\neg(A \vee B) \equiv \neg A \wedge \neg B$, $\neg(A \rightarrow B) \equiv A \wedge \neg B$
  A conjunctive formula is consistent (satisfiable) only if both of its component subformulas are satisfied.
- β - ***class*** contains disjunctive formulas:
  $A \vee B$, $\neg(A \wedge B) \equiv \neg A \vee \neg B$, $A \rightarrow B \equiv \neg A \vee B$
  A disjunctive formula is satisfied if at least one of its component subformulas is satisfiable.
- γ - ***class*** contains universal formulas:
  $(\forall x)A(x)$, $\neg(\exists x)A(x) \equiv (\forall x)\neg A(x)$
- δ - ***class*** contains existential formulas:
  $(\exists x)A(x)$, $\neg(\forall x)A(x) \equiv (\exists x)\neg A(x)$

According to these classes there are four ***decomposition rules***, symbolized graphically as follows:

α - ***rules***			β - ***rules***		
$A \wedge B$	$\neg(A \vee B)$	$\neg(A \rightarrow B)$	$A \vee B$	$\neg(A \wedge B)$	$A \rightarrow B$
$\mid$	$\mid$	$\mid$	$/ \backslash$	$/ \quad \backslash$	$/ \quad \backslash$
A	$\neg A$	A	$A \quad B$	$\neg A \quad \neg B$	$\neg A \quad B$
$\mid$	$\mid$	$\mid$			
B	$\neg B$	$\neg B$			

δ - ***rules***		
$(\exists x)A(x)$	$\neg(\forall x)A(x)$	c is a new parameter on the branch.
$\mid$	$\mid$	
$A(c)$	$\neg A(c)$	



$c_1, c_2, \ldots, c_n$ are all the parameters (constants) on that branch and they are used for the instantiation of A. If there is no constant on that branch, a new constant is introduced and used for instantiation.

Remarks:

- α ***-rule*** decomposes a conjunctive formula by adding on the same branch its component subformulas.
  This rule can be generalized for conjunctive formulas having more than two components: all its component subformulas are added on the same branch.
- β ***-rule*** decomposes a disjunctive formula by adding two new successors on two new branches.
- δ ***-rule*** introduces a new constant used for instantiation and the corresponding instance is added on the same branch.
- γ ***-rule*** adds on the same branch all the instances of the universal formula using for instantiation all the constants on that branch and also adds a copy of the decomposed universal formula. In case of new constants introduced in the decomposition process, this copy will be used for new instantiations.

Definition 3.1. The construction of a semantic tableau

To a propositional/predicate formula U we can associate a **semantic tableau**, which is a binary tree having formulas in its nodes and it is built as follows:

1. the root of the tree is labeled with the initial formula;
2. every branch of the tree which contains a formula will be extended with a subtree according to the decomposition rule specific to its class;
3. the extension of a branch stops in the following cases:
   a) if that branch contains a formula and its negation; the branch is marked as closed using the symbol $\otimes$;
   b) if all the formulas on that branch are already decomposed or if by decomposing the formulas which are not decomposed yet, no new formulas are obtained.

Definition 3.2.

1. A ***branch*** of the semantic tableau is called ***closed*** (marked by $\otimes$) if it contains a formula and its negation. Otherwise the ***branch*** is called ***open*** (marked by the symbol $\odot$).
2. A ***branch*** is called ***complete*** if it is closed or all the formulas on that branch are already decomposed.
3. A ***semantic tableau*** is called ***closed*** if all its branches are closed.
4. If a semantic tableau has at least one open branch, the ***tableau*** is called ***open***.
5. A ***semantic tableau*** is called ***complete*** if all its branches are complete.

Remarks:

- A closed branch symbolizes inconsistency among the formulas on that branch.
- The set of formulas belonging to a complete and open branch is consistent, meaning that it has a model.

The process of building a semantic tableau is not deterministic because at some steps there is a choice of the branch to work with and a choice of which formula to be decomposed. Thus, to a formula we can associate more than one semantic tableau and all of them are logically equivalent.

Recommendations to obtain the simplest semantic tableau (binary tree), with fewest branches:

- apply α - rules and δ - rules which keep one branch before β - rules which make a branching;
- apply δ - rules (introduction of new constants) before γ - rules which use all the constants on that branch.

All the formulas belonging to a branch are connected by conjunction and all the branches are connected by disjunction. A semantic tableau is the disjunction of all its branches and it is a graphical representation of a disjunctive normal form (DNF) of the formula from the root node.

A consistent (satisfiable) formula has associated a complete and open semantic tableau. Each open branch provides a partial model of the formula, which covers one or more models (complete models).

If from the semantic tableau of the formula $U(p_1, p_2, ..., p_n)$, 2^n distinct models are obtained, then the formula U is a tautology.

An inconsistent formula has associated a closed semantic tableau. All the branches of the semantic tableau are closed, meaning that there is no model of that formula.

The anti-models of a formula U, are the models of $\neg U$ and are provided by the open branches of the semantic tableau associated to $\neg U$.

From the previous assertions we conclude that it is easier to check if a formula U is a tautology, building the semantic tableau for $\neg U$ and checking if it is closed or not, than finding all the models of U from its semantic tableau.

Finding the models of a formula:

- For a *propositional formula* U, all its models are provided by all the open branches of the semantic tableau associated to U. Assigning the truth value T to all the literals belonging to an open branch, a partial/complete model of U is obtained. If a propositional variable is not on that branch, we can assign to it either T or F, obtaining more models.
- An open branch of the semantic tableau associated to the *predicate formula* U provides a partial/complete model of U with the minimum number of elements in the domain of interpretation. Such a model $I =< D, m >$ is a 'generic' one and it is built as follows:
  - the domain of interpretation D contains all the constants on that branch;
  - the value T is assigned to all the literals on that branch:
    if $c_1,...,c_n \in D$, $p \in P_n$ (P is an n - ary predicate) and
    $$\begin{cases} \text{a) } P(c_1,...,c_n) \text{ belongs to the branch then } m(P)(c_1,...,c_n) = T \\ \text{b) } \neg P(c_1,...,c_n) \text{ belongs to the branch then } m(P)(c_1,...,c_n) = F \end{cases}$$
  - based on this generic model, more concrete models can be obtained.

The semantic tableaux method is used as a refutation proof method to solve the decision problems in propositional/predicate logic, according to the following theorems.

Theorem 3.1. Soundness and completeness of the semantic tableaux method
A propositional/predicate formula V is a theorem (tautology) if and only if there is a closed semantic tableau associated to $\neg V$.

Theorem 3.2.
Let $U_1, U_2, \ldots, U_n, V$ be propositional/predicate formulas.
$U_1, U_2, \ldots, U_n \vdash V$ (equivalent to $U_1, U_2, \ldots, U_n \models V$) *if and only if* there is a closed semantic tableau associated to the formula $U_1 \wedge U_2 \wedge \ldots \wedge U_n \wedge \neg V$.

In propositional logic, which is decidable, the semantic tableau associated to a formula is always finite, therefore we can always decide if a formula is a tautology or not.
In predicate logic the semantic tableau can be infinite, caused by the combination of γ -rules and δ -rules.

The *predicate logic is not decidable, it is only semi-decidable*:

- if the semantic tableau associated to the formula $\neg V$ is finite, we can decide if the formula V is a tautology (closed tableau for $\neg V$) or not (open tableau for $\neg V$).
- if the semantic tableau of the formula $\neg V$ is infinite we cannot decide upon the validity of V.

Example 3.1.

Build two different semantic tableaux for the propositional formula:

$U = (p \vee q) \wedge \neg(q \to r) \wedge (p \to q \wedge r)$.

Tableau 1

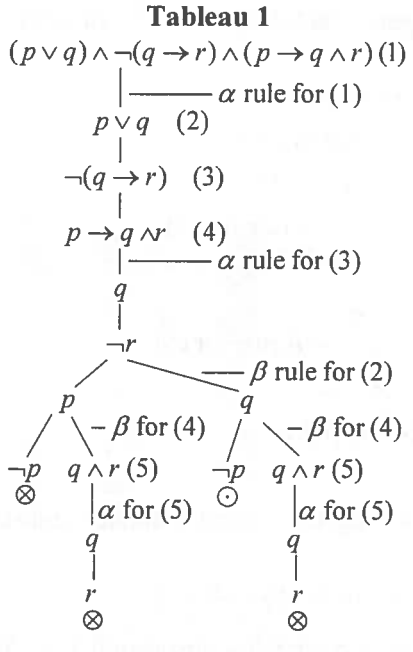
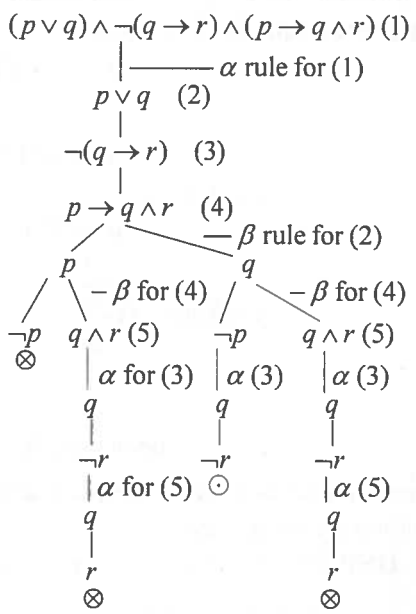


Tableau 2



Note that Tableau 1 is simpler (with fewer nodes on the branches) than Tableau 2, due to the fact that all the conjunctive subformulas were decomposed before the disjunctive subformulas (α - rules applied before β - rules).

These two semantic tableaux are equivalent and they represent graphically disjunctive normal forms logically equivalent to U.

From Tableau 1 we obtain:

$$\text{DNF}(U) = (q \wedge \neg r \wedge \underline{p} \wedge \underline{\neg p}) \vee (q \wedge \underline{\neg r} \wedge p \wedge q \wedge \underline{r}) \vee$$
$$\vee (q \wedge \neg r \wedge q \wedge \neg p) \vee (q \wedge \underline{\neg r} \wedge q \wedge q \wedge \underline{r})$$

From Tableau 2 we obtain:

$$\text{DNF}(U) = (\underline{p} \wedge \underline{\neg p}) \vee (p \wedge q \wedge \underline{\neg r} \wedge q \wedge \underline{r}) \vee$$
$$\vee (q \wedge \neg p \wedge q \wedge \neg r) \vee (q \wedge q \wedge \underline{\neg r} \wedge q \wedge \underline{r})$$

A simplified disjunctive normal form of U, obtained by eliminating the inconsistent cubes (they contain a pair of opposite literals - underlined) corresponding to the closed branches, is:

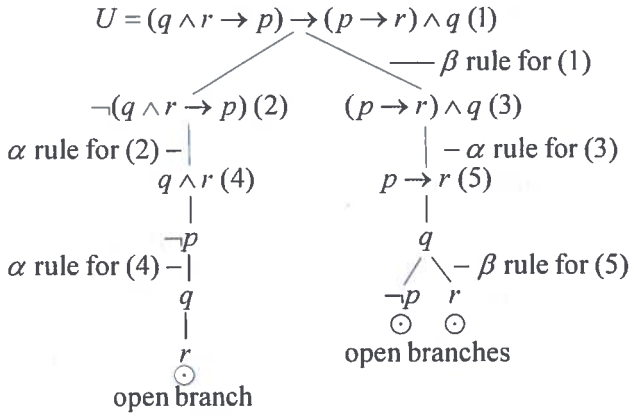
$\mathrm{DNF}(U) = \neg p \wedge q \wedge \neg r$.

This cube corresponds to the only open branch of both semantic tableaux and provides the unique model i of *the contingent formula* U:

$i : \{p,q,r\} \to \{T,F\}, i(p) = F, i(q) = T, i(r) = F$ and $i(U) = T$

Example 3.2.

Build a semantic tableau for $U = (q \wedge r \to p) \to (p \to r) \wedge q$ and decide what kind of formula (consistent, inconsistent, contingent, tautology) is U. In case of consistency find all its models.



The semantic tableau is complete and open, having three open branches and thus the formula is consistent.

$\mathrm{DNF}(U) = (\neg p \wedge q \wedge r) \vee (q \wedge \neg p) \vee (q \wedge r) \equiv (q \wedge \neg p) \vee (q \wedge r)$

The simplified form of DNF was obtained by applying the absorption law. The left-most branch, corresponding to the first cube, is covered by the other two branches.

The branch represented by the cube $q \wedge \neg p$ provides a partial model (the propositional variable r is missing) that covers two complete models of U:

$i_1 : \{p,q,r\} \to \{T,F\}, i_1(p) = F, i_1(q) = T, i_1(r) = T$

$i_2 : \{p,q,r\} \to \{T,F\}, i_2(p) = F, i_2(q) = T, i_2(r) = F$

The branch represented by the cube $q \wedge r$ provides the following models of U:

$i_3 : \{p,q,r\} \to \{T,F\}, i_3(p) = T, i_3(q) = T, i_3(r) = T$

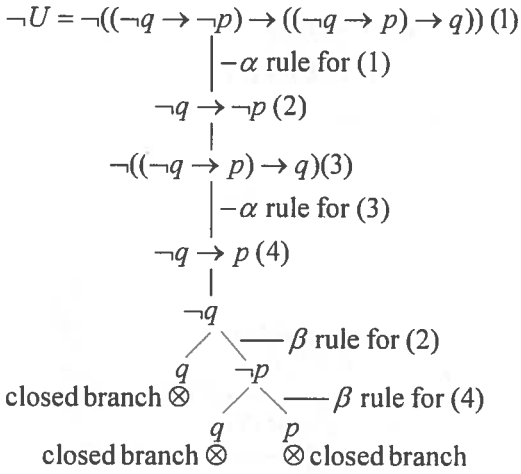
$i_4 : \{p,q,r\} \to \{T,F\}, i_4(p) = F, i_4(q) = T, i_4(r) = T$

Because $i_1 = i_4$, the formula U has three models: i_1, i_2 and i_3. $i_1(U) = i_2(U) = i_3(U) = T$.

Having at least one model, U is a consistent formula, but it is not a tautology (does not have 8 models), therefore U *is a contingent formula.*

Example 3.3.
Using the semantic tableaux method, prove that the formula $U=(\neg q\to\neg p)\to((\neg q\to p)\to q)$ is a tautology.

Theorem 3.1 is used, applying the semantic tableaux method as a refutation proof method. We build a semantic tableau for $\neg U$.



All three branches of the semantic tableau are closed, containing pairs of opposite literals: $(\neg q,q)$, $(\neg q,q)$, $(\neg p,p)$, therefore the formula $\neg U$ has no models because it is an inconsistent formula.
According to the theorem of soundness and completeness of this method we conclude that U is a tautology.
The semantic tableau corresponds to

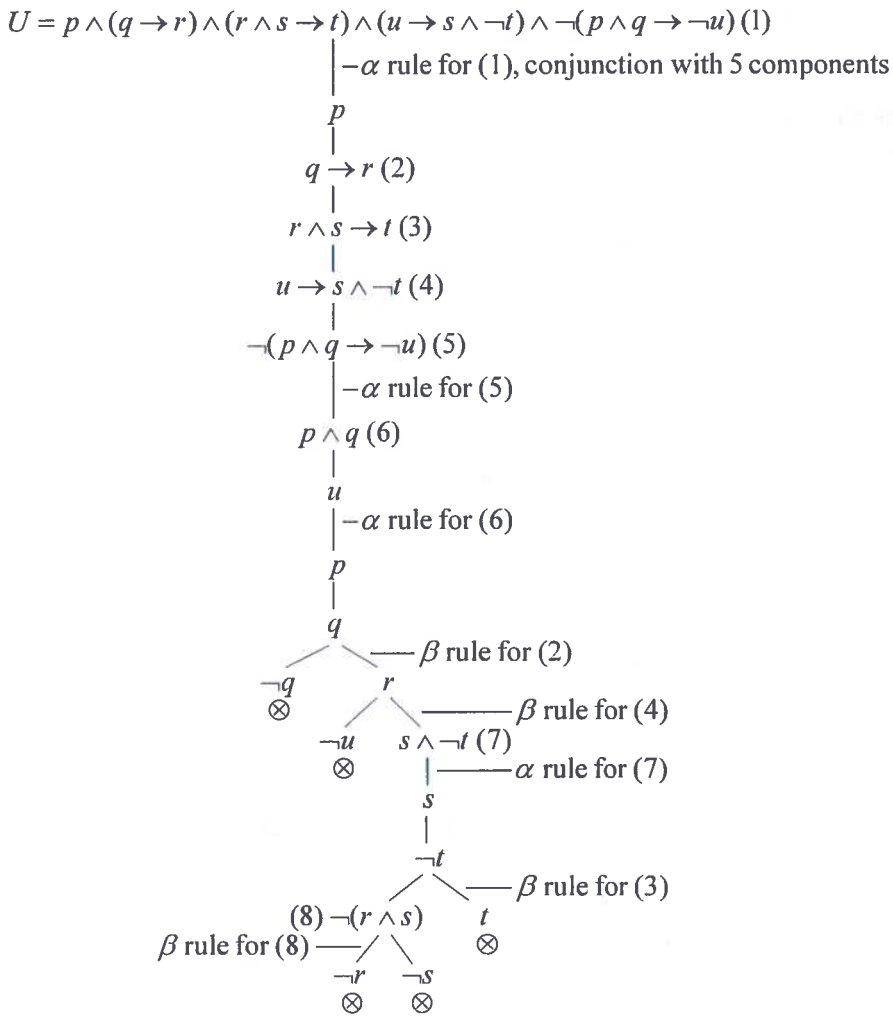
$$\mathrm{DNF}(\neg U)=(q\wedge\neg q)\vee(q\wedge\neg p\wedge\neg q)\vee(p\wedge\neg p\wedge\neg q).$$

Each cube represents a branch of the binary tree.
All the cubes are inconsistent, so $\neg U$ is an inconsistent formula and thus by contradiction U *is a tautology.*

Example 3.4.
Using the semantic tableaux method check whether the following logical consequence holds or not.

$$p\wedge(q\to r),r\wedge s\to t,u\to s\wedge\neg t\models p\wedge q\to\neg u.$$

We apply Theorem 3.2 and we build the semantic tableau associated to the formula U obtained as the conjunction of all the premises and the negation of the conclusion.



The semantic tableau associated to U is closed (with five closed branches), therefore U is an inconsistent formula and according to Theorem 3.2 we have proved by contradiction that *the following logical consequence holds*:

$$p \wedge (q \to r), r \wedge s \to t, u \to s \wedge \neg t \models p \wedge q \to \neg u .$$

Example 3.5.
Check if the formula $U = (\exists x)A(x) \to (\forall x)A(x)$ is a tautology in first-order logic.

We prove by contradiction, so we negate what we have to prove.
The semantic tableau associated to the formula $\neg U$ is represented by the following binary tree.

$\neg U = \neg((\exists x)A(x) \to (\forall x)A(x))$ (1)

α rule for (1)

$(\exists x)A(x)$ (2)

$\neg((\forall x)A(x))$ (3)

δ rule for (2), a - new constant

$A(a)$

δ rule for (3), b - new constant

$\neg A(b)$

$\odot$ open branch

The semantic tableau is complete and open, therefore $\neg U$ is consistent having at least one model which is an anti-model of U, so the formula U *is not a tautology.*
A generic model of $\neg U$ that is an anti-model of U, is the interpretation $I =< D, m >$ where $D = \{a, b\}$ is the domain of interpretation and contains all the constants belonging to the open branch.
To the predicate symbol A we assign the unary predicate $m(A)$ according to the literals belonging to the open branch as follows:
$m(A) : \{a, b\} \to \{T, F\}$, $m(A)(a) = T$ and $m(A)(b) = F$.

$v^I(\neg U) = T$ and $v^I(U) = F$

Example 3.6.
Check the validity of the predicate formula:

$U = (\exists x)A(x) \wedge (\exists x)B(x) \to (\exists x)(A(x) \wedge B(x))$

The semantic tableau associated to $\neg U$ is complete (there are no formulas such that by decomposing them new formulas which do not exist on that branch are obtained) and open. There are two closed branches and one open branch.

The open branch indicates that the formula $\neg U$ is consistent, thus U *is not valid.* The model of $\neg U$ provided by the open branch is an interpretation which evaluates the formula $\neg U$ as true. A model of $\neg U$ is an anti-model of U and it is obtained from the open branch as follows:

$I =< D, m >$,

$D = \{a, b\}$ - domain containing all the constants from the open branch

$m(A) : D \to \{T, F\}$ and $m(B) : D \to \{T, F\}$

$m(A)(a) = T$ because $A(a)$ belongs to the branch

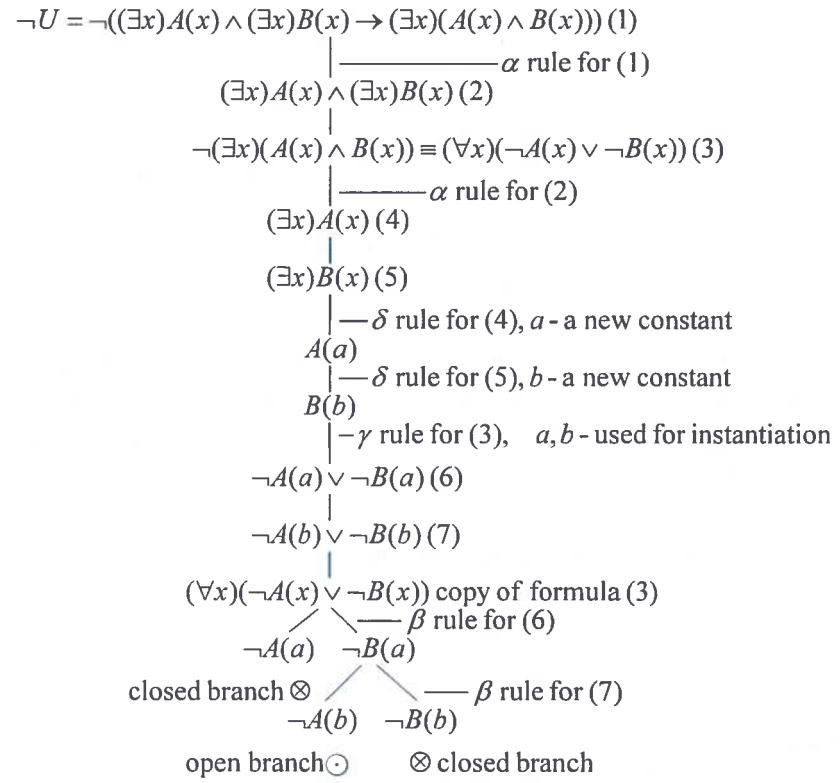
$m(A)(b) = F$ because $\neg A(b)$ belongs to the branch

$m(B)(a) = F$ because $\neg B(a)$ belongs to the branch

$m(B)(b) = T$ because $B(b)$ belongs to the branch

$v^I(\neg U) = T$ and $v^I(U) = F$

The binary tree corresponding to the semantic tableau of $\neg U$ is depicted below:



The copy of the formula (3), by decomposition, does not generate new formulas, because no new constants were introduced from the previous instantiation of (3).
An open branch provides a model with the minimum number of elements (the constants on that branch) in the domain of interpretation.

I is a 'generic' anti-model of U and based on it, a concrete anti-model I_1 can be obtained: $I_1 =< D_1, m_1 >, D_1 = \{4,5\}$ - domain

$m_1(A) : \{4,5\} \to \{T, F\}, m_1(A)(x) =$ 'x is an even number',

$$m_1(A)(4) = T, m_1(A)(5) = F$$

$m_1(B) : \{4,5\} \to \{T, F\}, m_1(B)(x) =$ 'x is an odd number',

$$m_1(B)(4) = F, m_1(B)(5) = T$$

$v^{I_1}(\neg U) = T$ and $v^{I_1}(U) = F$

The evaluation of the formula U under the interpretation I_1 is:

$$v^{I_1}(U) = (\exists x)_{x\in\{4,5\}}\text{'}x \textit{ is even}\text{'} \wedge (\exists x)_{x\in\{4,5\}}\text{'}x \textit{ is odd}\text{'} \to$$

$$\to (\exists x)_{x\in\{4,5\}}\text{'}x \textit{ is even and } x \textit{ is odd}\text{'} = T \wedge T \to F = F$$

Example 3.7.

Check the distributivity property of the universal quantifier over implication:

$(\forall x)(P(x) \to Q(x)) \equiv (\forall x)P(x) \to (\forall x)Q(x)$

We have that:

$(\forall x)(P(x) \to Q(x)) \equiv (\forall x)P(x) \to (\forall x)Q(x)$ if and only if

$\models (\forall x)(P(x) \to Q(x)) \leftrightarrow ((\forall x)P(x) \to (\forall x)Q(x))$ if and only if

$\models U_1$ and $\models U_2$ where:

$U_1 = (\forall x)(P(x) \to Q(x)) \to ((\forall x)P(x) \to (\forall x)Q(x))$ and

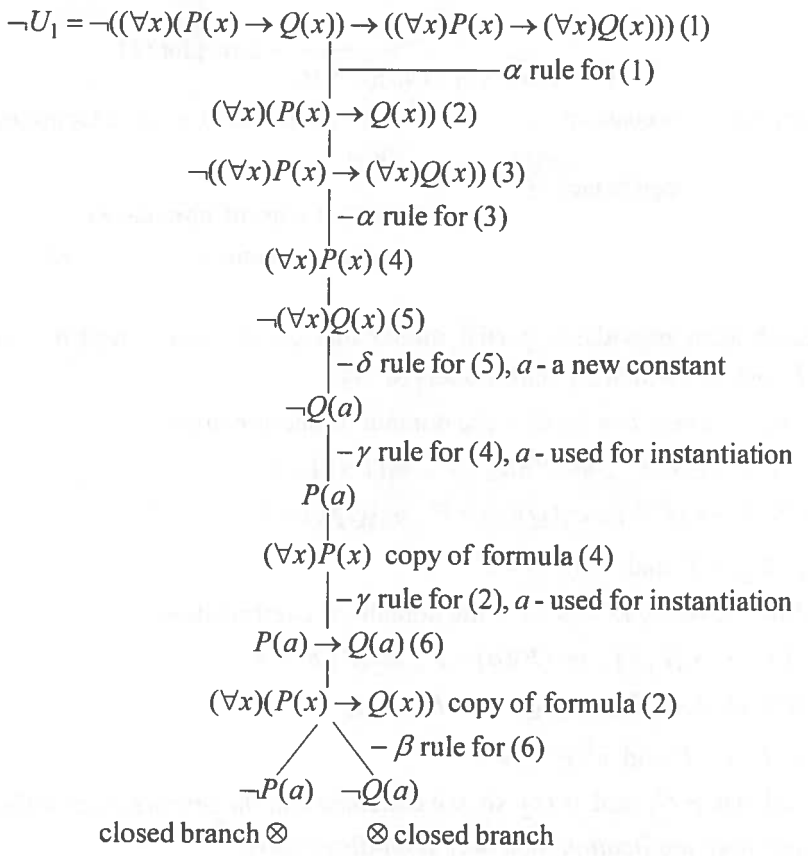
$U_2 = ((\forall x)P(x) \to (\forall x)Q(x)) \to (\forall x)(P(x) \to Q(x))$.

We shall prove that U_1 is a tautology and U_2 is not valid building the semantic tableaux of $\neg U_1$ and $\neg U_2$.

The following theoretic result is used:

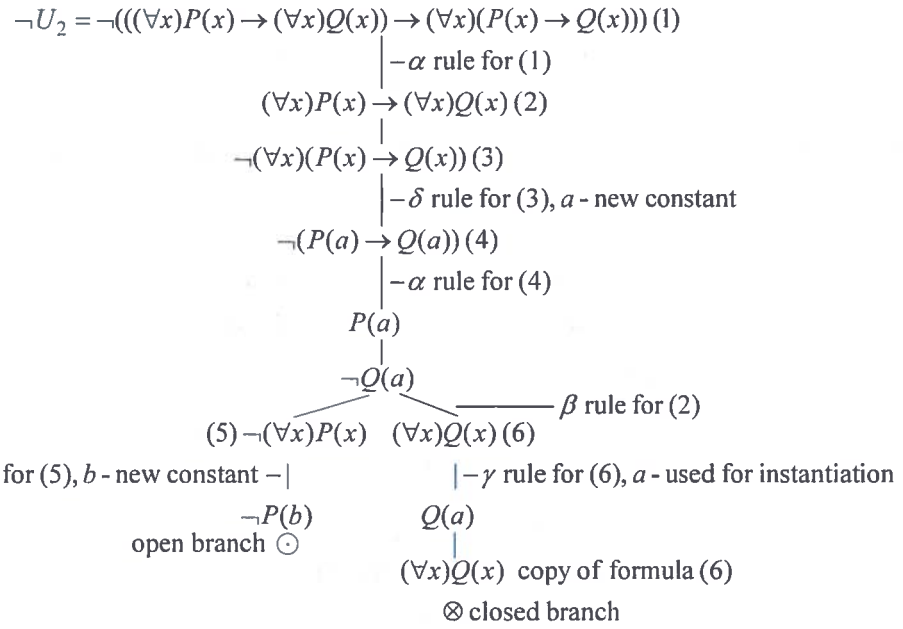
$\models U$ if and only if $\neg U$ has a closed semantic tableau.

The semantic tableau associated to $\neg U_1$ is represented graphically as follows:



The semantic tableau of $\neg U_1$ is closed, with two closed branches containing pairs of opposite literals: $(P(a), \neg P(a))$ and $(Q(a), \neg Q(a))$, therefore $\neg U_1$ is inconsistent and U_1 is valid.

The semantic tableau of $\neg U_2$, depicted below, is finite, complete and open, with one open branch and one closed branch, so $\neg U_2$ is consistent and U_2 is not a tautolog.



The open branch provides a partial model that covers two complete models of $\neg U_2$: I_1 and I_2, which are anti-models of U_2.

$I_1 = < D, m_1 >$, where $D = \{a, b\}$ – the domain of interpretation

$m_1(P): D \to \{T, F\}$, $m_1(P)(a) = T$, $m_1(P)(b) = F$,

$m_1(Q): D \to \{T, F\}$, $m_1(Q)(a) = F$, $m_1(Q)(b) = T$

$v^{I_1}(\neg U_2) = T$ and $v^{I_1}(U_2) = F$.

$I_2 = < D, m_2 >$, where $D = \{a, b\}$ – the domain of interpretation

$m_2(P): D \to \{T, F\}$, $m_2(P)(a) = T$, $m_2(P)(b) = F$,

$m_2(Q): D \to \{T, F\}$, $m_2(Q)(a) = F$, $m_2(Q)(b) = F$

$v^{I_2}(\neg U_2) = T$ and $v^{I_2}(U_2) = F$.

We proved that $\models U_1$ and $\not\models U_2$ so, we conclude that *the universal quantifier is not distributive over implication, it is only semi-distributive.*

Example 3.8.
Check the validity of the closed predicate formula:

$$U = (\forall x)(\forall y)P(x,y) \rightarrow (\exists x)(\exists y)P(x,y).$$

We negate the formula U and the semantic tableau associated to $\neg U$ is depicted below.

$\neg U = \neg((\forall x)(\forall y)P(x,y) \rightarrow (\exists x)(\exists y)P(x,y))$ (1)
| $-\alpha$ rule for (1)
$(\forall x)(\forall y)P(x,y)$ (2)
|
$\neg(\exists x)(\exists y)P(x,y)$ (3)
| $-\gamma$ rule for (2), a - new constant
$(\forall y)P(a,y)$ (4)
|
$(\forall x)(\forall y)P(x,y)$ copy of (2)
| $-\gamma$ rule for (4), a - is used for instantiation
$P(a,a)$
|
$(\forall y)P(a,y)$ copy of (4)
| $-\gamma$ rule for (3), a - is used for instantiation
$\neg(\exists y)P(a,y)$ (5)
|
$\neg(\exists x)(\exists y)P(x,y)$ copy of (3)
| $-\gamma$ rule for (5), a - is used for instantiation
$\neg P(a,a)$
|
$\neg(\exists y)P(a,y)$ copy of (5)
$\otimes$ closed branch

After applying α - rule, the formulas (2) and (3) are added on the branch. Both these formulas are universal formulas, we do not have yet constants to be used in instantiations, so γ - rule applied to (2) introduces a new constant ***a.*** All further applications of γ - rule in the decomposition process use this constant.

The semantic tableau of $\neg U$ is closed ($P(a,a)$ and $\neg P(a,a)$ are opposite literals), therefore U *is a valid formula* according to Theorem 3.1.

Example 3.9.
Let us consider the closed formula $U = (\exists y)(\forall x)P(x,y)$. We want to check the validity of U.

Using the 'reductio ad absurdum' principle we negate U in order to obtain a contradiction.

$\neg U = \neg((\exists y)(\forall x)P(x,y)) \equiv (\forall y)(\exists x)\neg P(x,y)$ (1)

$|-\gamma$ rule for (1), c_0 - new constant

$(\exists x)\neg P(x,c_0)$ (2)

$(\forall y)(\exists x)\neg P(x,y)$ (3) copy of formula (1)

$|-\delta$ rule for (2), c_1 - new constant

$\neg P(c_1,c_0)$

$|-\gamma$ rule for (3), uses the constant c_1, introduced at the previous instantiation of (2)

$(\exists x)\neg P(x,c_1)$ (4)

$(\forall y)(\exists x)\neg P(x,y)$ (5) copy of formula (1)

$|-\delta$ rule for (4), c_2 - new constant

$\neg P(c_2,c_1)$

$\vdots$

The semantic tableau associated to $\neg U$ is infinite, caused by the combination of γ-rules and δ-rules. The universal formula $(\forall y)(\exists x)\neg P(x,y)$ will be duplicated and it will generate at each application of γ-rule an existential formula of the form $(\exists x)\neg P(x,c_i)$, where c_i is the constant introduced by the existential formula at the previous step. The existential formula $(\exists x)\neg P(x,c_i)$ introduces a new constant c_{i+1} and the formula $\neg P(c_{i+1},c_i)$ will be added on the branch.
The unique branch of the binary tree contains only the literals:

$\neg P(c_1,c_0), \neg P(c_2,c_1), \neg P(c_3,c_2), \ldots$

Because the semantic tableau of $\neg U$ is infinite, we cannot decide upon the validity of U. Note that there is no possibility to obtain two opposite literals in order to close the tableau.

We prove that U is not valid (tautology) by finding an anti-model of it.
A well known interpretation which falsifies U is $I = < D, m >$, where the domain $D = \mathrm{N}$ is the set of all natural numbers and

$m(P) : \mathrm{N} \times \mathrm{N} \to \{T, F\}, m(P)(x,y) : "x < y"$.

This interpretation assigns to the binary predicate P the binary relation "$<$".
I is an anti-model of U.
Under the interpretation I, the formula $(\exists y)(\forall x)P(x,y)$ is translated as: "The biggest natural number exists", which is obvious false, and thus U *is not a tautology.*

3.2. A New Approach to the Semantic Tableaux Method

This section presents a new approach to the semantic tableaux method used at the implementation level [56].

Definition 3.3.
A ***branch*** is a set of literals connected by conjunction and the reunion of the branches is the ***semantic tableau*** (a set of sets of literals).

Definition 3.4.
The ***function TP***, which associates a semantic tableau to a formula, is defined as a mapping between sets of formulas and sets of sets of formulas. It is based on the decomposition rules: α, β, δ, γ as follows:

M is a set of formulas.

- $TP(M) = \{M\}$, if M is a set of literals.
- $TP(M) = TP(M' \cup \{p\})$, if $\neg\neg p \in M$ and $M' = M \setminus \{\neg\neg p\}$
- decomposition of conjunctive formulas (α-rule)
  $TP(M) = TP(M' \cup \{p\} \cup \{q\})$, if $p \wedge q \in M$ and $M' = M \setminus \{p \wedge q\}$
  $TP(M) = TP(M' \cup \{\neg p\} \cup \{\neg q\})$, if $\neg(p \vee q) \in M$ and $M' = M \setminus \{\neg(p \vee q)\}$
  $TP(M) = TP(M' \cup \{p\} \cup \{\neg q\})$, if $\neg(p \rightarrow q) \in M$ and $M' = M \setminus \{\neg(p \rightarrow q)\}$
- decomposition of disjunctive formulas (β-rule)
  $TP(M) = TP(M' \cup \{p\}) \cup TP(M' \cup \{q\})$, if $p \vee q \in M$ and $M' = M \setminus \{p \vee q\}$
  $TP(M) = TP(M' \cup \{\neg p\}) \cup TP(M' \cup \{\neg q\})$, if $\neg(p \wedge q) \in M$ and
  $M' = M \setminus \{\neg(p \wedge q)\}$
  $TP(M) = TP(M' \cup \{\neg p\}) \cup TP(M' \cup \{q\})$, if $p \rightarrow q \in M$ and
  $M' = M \setminus \{p \rightarrow q\}$
- decomposition of existential formulas (δ-rule)
  $TP(M) = TP(M' \cup \{A(c)\})$, if $(\exists x)A(x) \in M$, where c - new constant and
  $M' = M \setminus \{(\exists x)A(x)\}$
  $TP(M) = TP(M' \cup \{\neg A(c)\})$, if $\neg(\forall x)A(x) \in M$ where c - new constant and
  $M' = M \setminus \{\neg(\forall x)A(x)\}$
- decomposition of universal formulas (γ-rule)
  $TP(M) = TP(M \cup \{A(c) \mid c \in D\})$, if $(\forall x)A(x) \in M$
  $TP(M) = TP(M \cup \{\neg A(c) \mid c \in D\})$, if $\neg(\exists x)A(x) \in M$
  - where D is the domain of interpretation containing all the constants previously introduced. If $D = \varnothing$, then a new constant c is introduced and we have that $D = \{c\}$.

The semantic tableau $\cup_{i=1}^{m}\{\cup_{k=1}^{n_i}\{l_{i_k}\}\}$, where $l_{i_k}, i=\overline{1,m}, k=\overline{1,n_i}$ are literals, corresponds to a formula with the disjunctive normal form $\vee_{i=1}^{m}(\wedge_{k=1}^{n_i} l_{i_k})$. It has m branches and corresponds to the disjunction of its branches. The i-branch of the tableau is the set of literals: $\cup_{k=1}^{n_i}\{l_{i_k}\}$ and represents a cube, the conjunction of its literals.

Definition 3.5.

1. If a branch contains a literal and its negation, we say that the ***branch*** is ***closed***, otherwise the ***branch*** is ***open***.
2. If all the branches of a tableau are closed, the ***tableau*** is ***closed***, otherwise the ***tableau*** is ***open***.

Theorem 3.3. Soundness and completeness of *TP*

A propositional/predicate formula V is a tautology (theorem) if and only if $TP(\{\neg V\})$ is closed.

Theorem 3.4.

Let $U_1, U_2, \ldots, U_n, V$ be propositional/predicate formulas.

$U_1, U_2, \ldots, U_n \models V$ if and only if $TP(\{U_1, U_2, \ldots, U_n, \neg V\})$ is closed.

Remarks

- If the ***tableau*** associated to a set of formulas is **open** then the ***set of formulas*** is ***consistent.*** Each open branch provides a partial/complete model of the set of formulas.
- If the ***tableau*** associated to a set of formulas is **closed** then the ***set of formulas is inconsistent***

If the resulted semantic tableau is too complex we transform it into a ***simplified form*** by eliminating the closed branches (inconsistent sets) and the branches (sets) that contain another branch (set) from the tableau.

Definition 3.6.

Let M be a set of formulas. The simplified form of the semantic tableau $TP(M)$, denoted by $TS(M)$, is obtained as follows:

$TS(M) = TP(M) \setminus \{X \mid X \in TP(M), X \text{ is closed or } \exists Y \in TP(M), \varnothing \neq Y \subset X, Y \neq X\}$

Remarks:

1. $TS(M)$ is empty or contains only open branches.
2. Using the simplified form of a semantic tableau, the theoretic results from Theorem 3.3 and Theorem 3.4 can be written as follows:

Let $U_1, U_2, \ldots, U_n, V$ be propositional/predicate formulas.

a) V is a tautology (theorem) if and only if $TS(\{\neg V\}) = \varnothing$.

b) $U_1, U_2, \ldots, U_n \models V$ if and only if $TS(\{U_1, U_2, \ldots, U_n, \neg V\}) = \varnothing$

Example 3.10.

Check whether the following logical consequence holds using *TP* function.

$p \to q, \neg(q \to r) \to \neg p \models p \to r$.

We apply Theorem 3.4 and we begin with the set $M = \{p \to q, \neg(q \to r) \to \neg p, \neg(p \to r)\}$ containing the two hypotheses and the negation of the conclusion of the initial logical consequence:

$T_1 = TP(\{p \to q, \neg(q \to r) \to \neg p, \neg(p \to r)\})$

- decomposition of the conjunctive formula $\neg(p \to r)$:

  $T_1 = TP(\{p \to q, \neg(q \to r) \to \neg p, p, \neg r\})$

- decomposition of the disjunctive formula $p \to q$

  $T_1 = TP(\{\neg(q \to r) \to \neg p, p, \neg r, \neg p\}) \cup TP(\{\neg(q \to r) \to \neg p, p, \neg r, q\})$

- simplification: elimination of the closed subtableau:

  $TP(\{\neg(q \to r) \to \neg p, \underline{p}, \neg r, \underline{\neg p})\})$, contains a pair of opposite literals: $(p, \underline{\neg p})$

We obtain: $T_2 = TP(\{\neg(q \to r) \to \neg p, p, \neg r, q\})$

- decomposition of the disjunctive formula $\neg(q \to r) \to \neg p$

  $T_2 = TP(\{p, \neg r, q, q \to r\}) \cup TP(\{p, \neg r, q, \neg p\})$

- simplification: elimination of the subtableau $TP(\{\underline{p}, \neg r, q, \underline{\neg p}\})$ which is closed (contains a pair of opposite literals: $(p, \underline{\neg p})$)

We obtain: $T_3 = TP(\{p, \neg r, q, q \to r\})$

- decomposition of the disjunctive formula $q \to r$

  $T_3 = TP(\{p, \neg r, q, \neg q\}) \cup TP(\{p, \neg r, q, r\}) = \{\{p, \neg r, \underline{q}, \underline{\neg q}\}, \{p, \underline{\neg r}, q, \underline{r}\}\}$

- simplification: elimination of those two subtableau which are closed (the pairs of opposite literals are underlined),

$T_4 = \varnothing$

After applying decomposition rules and simplifications on the initial set M we obtained $TS(M) = T_4 = \varnothing$, thus *the logical consequence:*

$p \to q, \neg(q \to r) \to \neg p \models p \to r$ *holds.*

Example 3.11.

Using *TP* function check whether $U = (p \to q) \wedge p \wedge \neg(q \to r)$ is a consistent formula. If so, find the models of U.

We build: $T_1 = TP(\{(p \to q) \wedge p \wedge \neg(q \to r)\})$

- decomposition of the conjunctive formula $(p \to q) \wedge p \wedge \neg(q \to r)$
  $T_1 = TP(\{p \to q, p, \neg(q \to r)\})$
- decomposition of the conjunctive formula $\neg(q \to r)$
  $T_1 = TP(\{p \to q, p, q, \neg r\})$
- decomposition of the disjunctive formula $p \to q$
  $T_1 = TP(\{p, q, \neg r, \neg p\}) \cup TP(\{p, q, \neg r, q\})$
- simplification: elimination of the subtableau $TP(\{\underline{p}, q, \neg r, \underline{\neg p}\})$ which is closed (contains a pair of opposite literals: underlined)
  $T_2 = TP(\{p, q, \neg r, q\}) = \{\{p, q, \neg r\}\}$

The semantic tableau $T_2 = TS(\{(p \to q) \wedge p \wedge \neg(q \to r)\})$ is open because it contains an open branch, so U *is a consistent formula.*

According to the above semantic tableau the reduced disjunctive normal form of U is: $\mathrm{DNF}(U) = p \wedge q \wedge \neg r$.

U has a unique model:

$i : F_p \to \{T, F\}$ with $i(p) = T$, $i(q) = T$, $i(r) = F$ and $i(U) = T$.

Example 3.12.

Check the distributivity property of the existential quantifier $(\exists)$ over implication $(\to)$ using *TP* function.

The property is expressed by the following logical equivalence relation:

$((\exists x)A(x) \to (\exists x)B(x)) \equiv (\exists x)(A(x) \to B(x))$.

The problem is reduced to checking the validity of the formulas: U_1 and U_2, where:

$U_1 = ((\exists x)A(x) \to (\exists x)B(x)) \to (\exists x)(A(x) \to B(x))$

$U_2 = (\exists x)(A(x) \to B(x)) \to ((\exists x)A(x) \to (\exists x)B(x))$

We build $TP(\{\neg U_1\})$:

$T_1 = TP(\{\neg U_1\}) = TP(\{\neg(((\exists x)A(x) \to (\exists x)B(x)) \to (\exists x)(A(x) \to B(x)))\})$

- decomposition of the conjunctive formula: $\neg U_1$:
  $T_1 = TP(\{(\exists x)A(x) \to (\exists x)B(x), \neg(\exists x)(A(x) \to B(x))\})$
- decomposition of the disjunctive formula: $(\exists x)A(x) \to (\exists x)B(x)$:
  $T_1 = TP(\{\neg(\exists x)A(x), (\forall x)\neg(A(x) \to B(x))\}) \cup$
  $TP(\{(\exists x)B(x), (\forall x)\neg(A(x) \to B(x))\})$

- decomposition of the existential formula $(\exists x)B(x)$, $\boldsymbol{b}$ - a new constant, on the second branch and decomposition of the universal formula $\neg(\exists x)A(x) \equiv (\forall x)\neg A(x)$, a new constant $\boldsymbol{a}$ is introduced and used for instantiation, because there are no constants on the first branch:

$$T_1 = TP(\{\neg A(a), \neg(\exists x)A(x), (\forall x)\neg(A(x) \to B(x))\}) \cup$$
$$\cup TP(\{B(b), (\forall x)\neg(A(x) \to B(x))\})$$

- decomposition of the universal formula $(\forall x)\neg(A(x) \to B(x))$ on both branches, with the appropriate instantiations on each branch:

$$T_1 = TP(\{\neg A(a), (\forall x)\neg A(x), \neg(A(a) \to B(a)), (\forall x)\neg(A(x) \to B(x))\}) \cup$$
$$\cup TP(\{B(b), \neg(A(b) \to B(b)), (\forall x)\neg(A(x) \to B(x))\})$$

- decomposition of the conjunctive formulas on both branches:

$$T_1 = TP(\{\neg A(a), (\forall x)\neg A(x), A(a), \neg B(a), (\forall x)\neg(A(x) \to B(x))\}) \cup$$
$$\cup TP(\{B(b), A(b), \neg B(b), (\forall x)\neg(A(x) \to B(x))\}) =$$
$$= \{\{\underline{\neg A(a)}, \underline{A(a)}, \neg B(a)\}, \{\underline{B(b)}, A(b), \underline{\neg B(b)}\}\}$$

T_1 is a closed semantic tableau because it has two closed branches, each branch contains a pair of opposite literals (underlined). Therefore the formula $\neg U_1$ is inconsistent and U_1 *is valid.*

We shall prove that U_2 is not valid because $TP(\neg U_2)$ is a complete and open semantic tableau.

$$TP(\{\neg U_2\}) = TP(\{\neg((\exists x)(A(x) \to B(x)) \to ((\exists x)A(x) \to (\exists x)B(x)))\}) =$$
$$= TP(\{(\exists x)(A(x) \to B(x)), \neg((\exists x)A(x) \to (\exists x)B(x))\})$$

δ -rule is applied, a new constant $\boldsymbol{a}$ is introduced

$$= TP(\{A(a) \to B(a), \neg((\exists x)A(x) \to (\exists x)B(x))\}) =$$
$$= TP(\{A(a) \to B(a), (\exists x)A(x), \neg(\exists x)B(x)\})$$

γ -rule is applied, the constants $\boldsymbol{a}$ and $\boldsymbol{b}$ are used for instantiation

$$= TP(\{A(a) \to B(a), A(b), \neg(\exists x)B(x)\}) =$$

δ -rule is applied, a new constant $\boldsymbol{b}$ is introduced

$$= TP(\{A(a) \to B(a), A(b), \neg B(a), \neg B(b), \neg(\exists x)B(x)\}) =$$

β -rule is applied

$$= TP(\{\neg A(a), A(b), \neg B(a), \neg B(b), \neg(\exists x)B(x)\}) \cup$$
$$\cup TP(\{B(a), A(b), \neg B(a), \neg B(b), \neg(\exists x)B(x)\}) =$$
$$= \{\{\neg A(a), A(b), \neg B(a), \neg B(b), \neg(\exists x)B(x)\},$$
$$\{\underline{B(a)}, A(b), \underline{\neg B(a)}, \neg B(b), \neg(\exists x)B(x)\}\}$$

The final semantic tableau is complete and open, having one complete open branch (the first branch) and one closed branch ($\underline{B(a)}, \underline{\neg B(a)}$) is the pair of opposite

literals on the second branch). The formula $\neg U_2$ is not inconsistent, so U_2 *is not valid.*

We conclude that *the existential quantifier is only semi-distributive over implication:*

$\models ((\exists x)A(x) \rightarrow (\exists x)B(x)) \rightarrow (\exists x)(A(x) \rightarrow B(x))$ and

$\not\models (\exists x)(A(x) \rightarrow B(x)) \rightarrow ((\exists x)A(x) \rightarrow (\exists x)B(x))$

3.3. An Automated Theorem Prover for Propositional Logic, based on the Semantic Tableaux Method

The aim of the proposed propositional theorem prover is to check the consistency or inconsistency of a propositional formula/set of formulas and to provide a model in case of consistency.

This theorem prover is based on a modified version of the semantic tableaux method and it is described in paper [35]. We will use the same representation for a tableau as the function *TP*: a set of sets of literals, but the construction of the tableau is different.

A disadvantage of the definition of *TP* function described in the previous section is the fact that for a formula we can build many semantic tableaux, depending on the order of the application of the decomposition rules, and because of this it is hard to obtain a minimal form of the tableau.

The new idea is to construct the semantic tableau of a propositional formula from its postfix form. Traversing the postfix form from left to right, using a stack mechanism to memorize partial semantic tableaux (corresponding to the subformulas of the formula), and applying operations to the tableaux, the construction of the semantic tableau is very simple and efficient.

Data structures used to represent and manipulate propositional formulas

Postfix and prefix notations of arithmetic expressions are used internally by computers to manipulate and evaluate these expressions.

infix notation: 9+(8-2*3*5)/6+7+4*5

- operators are in between the corresponding operands;
- parentheses are necessary to impose the order of operations;

postfix notation: 9 8 2 3 * 5 * - 6 / + 7 + 4 5 * +

- operators are after the corresponding operands, no parentheses are needed;

prefix notation: + + + 9 / - 8 * * 2 3 5 6 7 * 4 5

- operators are before the corresponding operands, no parentheses are needed;

All three notations have the same order of the operands.

Traversing the postfix notation from left to right, using a stack mechanism to memorize partial results the expression is evaluated.

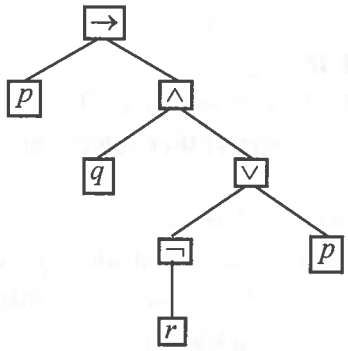
In the same manner, propositional formulas can be represented in prefix and postfix notations or as a binary tree corresponding to the decomposition of the formula in subformulas.

formula: $p \to q \wedge (\neg r \vee p)$ - infix notation

postfix notation: $p\ q\ r \neg p \vee \wedge \to$

prefix notation: $\to p\ \wedge q \vee \neg r\ p$

formula tree:



The root node contains the main connective of the formula. Each leaf node is labelled with a propositional variable and the other nodes contain connectives.
All these special representations help us to manipulate more efficiently the propositional formulas at the implementation level.

The following subalgorithm transforms a propositional formula into its postfix notation using a stack mechanism.

Subalgorithm **postfix_notation**(*formula, postfix*)
input parameter: *formula* – initial propositional formula,
output parameter: *postfix* – formula in postfix notation
// *stack* used to memorize connectives and open parentheses
// *formula, postfix* are represented as vectors
begin
  $j := 0$
  for $i = 1$, length(*formula*)
    if (*formula*[i] is a propositional_variable)
      then $j := j + 1$;
           postfix[j] := *formula*[i];
      else
        if (*formula*[i] ='(')
          then push(*stack*, '(')
          else

```
            if ( formula[i] =')' )
              then // extract from stack and add to postfix all the connectives
                                                          // until '('
                do
                  x := pop(stack)
                  if( x is a connective)
                    then  j := j + 1 ;
                          postfix[j] := x ;
                  end_if
                until (stack is empty or x ='(' )
                if (stack is empty) then return("error")
                end_if
              else //  formula[i]  is a connective
                // extract from stack and add to postfix all the connectives with
                    // a higher or equal priority than the connective  formula[i]
                oper := top_stack(stack);
                     // oper is the connective from the top of stack
                while (stack is not empty and pr(oper) >= pr( formula[i] ))
                  // pr( x ) returns the priority of the connective  x
                    oper := pop(stack)
                    j := j + 1 ;
                    postfix[j] := oper ;
                    oper := top_stack(stack)
                end_while
                push(stack, formula[i] )
          end_if
        end_if
      end_if
    end_for
    // all the remaining connectives from stack are extracted and added to postfix
    while (stack is not empty)
      oper := pop(stack)
      j := j + 1 ;
      postfix[j] := oper ;
    end_while
end
```

The formula tree can be built from the postfix form of the formula, using a stack mechanism to memorize nodes addresses according to the subalgorithm below.

```
Subalgorithm build_tree_from_postfix(postfix, adr_tree)
input parameter: postfix – formula in postfix notation
output parameter: adr_tree – the address of the root tree
// stack_adr is used to memorize nodes addresses
begin
  for i = 1, length(postfix)
    if ( postfix[i] is a propositional_variable )
      then
        adr_node := build_node ( postfix[i], NULL, NULL);
        push( stack_adr, adr_node);
      else // postfix[i] is a connective
        adr_right := pop(stack_adr);
        if ( postfix[i] = '¬' ) // the negation connective
          then adr_left := NULL;
          else adr_left := pop(stack_adr);
        end_if
        adr_node := build_node ( postfix[i], adr_left, adr_right)
        push (stack_adr, adr_node);
    end_if
  end_for
  adr_tree := pop(stack_adr);
end
```

3.4. *Considerations for implementation*

A semantic tableau is represented as a set of sets of literals. The logical operations (connectives) of a formula are transformed into corresponding operations on semantic tableaux according to the definitions bellow [35].

Definition 3.7.
Let us denote by $Tsem(U)$ and $Tsem(V)$ the semantic tableaux associated to the propositional formulas U and V.

$Tsem(l) = \{\{l\}\}$, where 'l' is a propositional literal

$Tsem(\neg U) = \neg Tsem(U)$,

$Tsem(U \wedge V) = Tsem(U) \wedge Tsem(V)$,

$Tsem(U \vee V) = Tsem(U) \vee Tsem(V)$,

$Tsem(U \rightarrow V) = Tsem(U) \rightarrow Tsem(V)$,

$Tsem(U \leftrightarrow V) = Tsem(U) \leftrightarrow Tsem(V)$.

The simplification of a semantic tableau, in order to obtain a minimal form, corresponds to the elimination of closed sets of literals, and sets containing other sets from the tableau. An empty simplified semantic tableau corresponds to a closed tableau. If we do not apply the simplification, then the final semantic tableau corresponds to the disjunctive normal form of the initial formula.

Definition 3.8.

Let $T_1 = \cup_{i=1}^{N} \{\cup_{k=1}^{n_i} \{a_{ik}\}\}$ and $T_2 = \cup_{j=1}^{M} \{\cup_{k=1}^{m_j} \{b_{jk}\}\}$ be two semantic tableaux represented using *TP* function as sets of sets of literals. The following relations provide the application of the logical operations: $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$, on semantic tableaux:

$$\neg T_1 = \{\{\neg x_1, ..., \neg x_N\} \mid x_i \in \cup_{k=1}^{n_i} \{a_{ik}\}, i = 1, ..., N\}$$

$$T_1 \vee T_2 = \{\cup_{k=1}^{n_i} \{a_{ik}\} \mid i = 1, ..., N\} \cup \{\cup_{k=1}^{m_j} \{b_{jk}\} \mid j = 1, ..., M\}$$

$$T_1 \wedge T_2 = \{\cup_{k=1}^{n_i} \{a_{ik}\} \cup \cup_{k=1}^{m_j} \{b_{jk}\} \mid i = 1, ..., N, j = 1, ..., M\}$$

$$T_1 \rightarrow T_2 = \neg T_1 \vee T_2$$

$$T_1 \leftrightarrow T_2 = (T_1 \rightarrow T_2) \wedge (T_2 \rightarrow T_1)$$

The above operations permit the construction of the semantic tableau of a formula from the semantic tableaux of its subformulas.

Example 3.13.

Let $T_1 = \{\{p, \neg q, r\}, \{\neg p\}, \{q, \neg r\}\}$ and $T_2 = \{\{s, r\}, \{t\}\}$ be two semantic tableaux represented using *TP* function. We apply the previous definition.

$$\neg T_1 = \{\{\neg p, p, \neg q\}, \{\neg p, p, r\}, \{q, p, \neg q\}, \{q, p, r\}, \{\neg r, p, \neg q\}, \{\neg r, p, r\}\}$$

The simplified form of T_1 obtained by eliminating the inconsistent sets (closed branches containing a pair of opposite literals) is $TS = \{\{q, p, r\}, \{\neg r, p, \neg q\}\}$

$$T_1 \vee T_2 = \{\{p, \neg q, r\}, \{\neg p\}, \{q, \neg r\}, \{s, r\}, \{t\}\}$$

$$T_1 \wedge T_2 = \{\{p, \neg q, r, s, r\}, \{p, \neg q, r, t\}, \{\neg p, s, r\}, \{\neg p, t\}, \{q, \neg r, s, r\}, \{q, \neg r, t\}\}$$

The simplified form of $T_1 \wedge T_2$ is

$$TS = \{\{p, \neg q, r, s\}, \{p, \neg q, r, t\}, \{\neg p, s, r\}, \{\neg p, t\}, \{q, \neg r, t\}\}$$

$$T_1 \rightarrow T_2 = \neg T_1 \vee T_2 = \{\{q, p, r\}, \{\neg r, p, \neg q\}\} \vee \{\{s, r\}, \{t\}\} =$$

$$= \{\{q, p, r\}, \{\neg r, p, \neg q\}, \{s, r\}, \{t\}\}$$

Example 3.14.

The formula $U = \neg(p \wedge q) \vee r \wedge \neg s$ has its postfix notation: $pq \wedge \neg rs \neg \wedge \vee$.

The semantic tableau associated to U: $Tsem(U)$ is built step by step traversing the postfix form from left to right and using a stack which memorizes the partial semantic tableaux (see the next algorithm).

Symbol from postfix form	postfix notation: $pq \wedge \neg rs \neg \wedge \vee$ **partial semantic tableaux**	**stack**
'p'	$T_1 = Tsem(p) = \{\{p\}\}$	(T_1)
'q'	$T_2 = Tsem(q) = \{\{q\}\}$	(T_2, T_1)
'$\wedge$'	$T_3 = T_1 \wedge T_2 = \{\{p\}\} \wedge \{\{q\}\} = \{\{p,q\}\}$	(T_3)
'$\neg$'	$T_4 = \neg T_3 = \neg\{\{p,q\}\} = \{\{\neg p\},\{\neg q\}\}$	(T_4)
'r'	$T_5 = Tsem(r) = \{\{r\}\}$	(T_5, T_4)
's'	$T_6 = Tsem(s) = \{\{s\}\}$	(T_6, T_5, T_4)
'$\neg$'	$T_7 = \neg T_6 = \neg\{\{s\}\} = \{\{\neg s\}\}$	(T_7, T_5, T_4)
'$\wedge$'	$T_8 = T_5 \wedge T_7 = \{\{r\}\} \wedge \{\{\neg s\}\} = \{\{r, \neg s\}\}$	(T_8, T_4)
'$\vee$'	$T_9 = T_4 \vee T_8 = \{\{\neg p\},\{\neg q\}\} \vee \{\{r, \neg s\}\}$ $= \{\{\neg p\},\{\neg q\},\{r, \neg s\}\}$	(T_9)

$Tsem(U) = T_9 = \{\{\neg p\},\{\neg q\},\{r, \neg s\}\}$

DNF(U)= $\neg p \vee \neg q \vee (r \wedge \neg s)$

This new algorithm, used to build a semantic tableau for a propositional formula from its postfix notation, is more efficient than the algorithms which use the decomposition rules.

Subalgorithm **construction_semantic_tableau (postfix, tab)**
input parameter: postfix – postfix notation of the initial formula,
output parameter: tab – the semantic tableau
// **stack** is used to memorize the partial semantic tableaux
begin
  for $i = 0$, length(**postfix**)
    if (*postfix*[i] **is a propositional_variable**)
      then
        $t := Tsem(postfix[i])$ // $Tsem(postfix[i]) := \{\{postfix[i]\}\}$
        push(**stack**, t) //the semantic tableau t is added to the *stack*
      else

```
        if ( postfix[i] is "¬")
         then
            t := pop(stack);
            t_l := ¬t ;
            t := simplify( t_l );
            push(stack, t )
          else // postfix[i] is one of the binary connectives { ∧,∨,→,↔ }
            t_r := pop(stack) // t_r is the tableau corresponding to the right
                              // subformula (operand) of the connective postfix[i]
            t_l := pop(stack) // t_l is the tableau corresponding to the left
                              // subformula (operand) of the connective postfix[i]
            t := simplify ( t_l postfix[i]t_r ) // tableau for t_l postfix[i]t_r is built
                                                                  and simplified
            push(stack, t ) // the simplified tableau is added to the stack
        end_if
    end_if
  end_for
  tab := pop (stack)
  // the tableau extracted from the stack is the final semantic tableau
  // corresponding to the initial formula provided by its postfix notation
end
```

The following algorithm implements the semantic tableaux proof method and can be used to solve three tasks:

1. to check the validity of a propositional formula V;
2. to check the consistency and find the models of a propositional formula V;
3. to check if a propositional formula V is a logical consequence of the set of hypotheses: $U_1,...,U_n$.

```
Algorithm Semantic_tableaux_method
input: V,U_1,U_2,...,U_n - well-formed propositional formulas
       opt ∈ {1,2,3} - option for the task to be solved
output: a message depending on opt
begin
  if ( opt = 1 ) then //checking_tautology
          postfix_notation( ¬V , postfix)
          construction_semantic_tableau(postfix, tab)
          if (tab is closed) then
              write "¬V is inconsistent and V is a tautology"
```

```
            else
               write"¬V is consistent and V is not a tautology"
            end_if
   end_if
   if (opt = 2) then //checking_consistent_formula
            postfix_notation(V, postfix)
            construction_semantic_tableau(postfix, tab)
            if (tab is closed) then write "V is inconsistent"
               else
               write "V is consistent"
               constructFormulaModels(tab)
            end_if
   end_if
   if (opt = 3) then //checking_logical_consequence
            C = U1 ∧ U2 ∧ ... ∧ Un ∧ ¬V
            postfix_notation(C, postfix)
            construction_semantic_tableau(postfix, tab)
            if (tab is closed) then
               write "V is a logical consequence of U1,U2,...,Un"
              else write "V is not a logical consequence of U1,U2,...,Un"
            end_if
    end_if
end
```

3.5. Exercises

Exercise 3.1.

Using the semantic tableaux method (the binary tree and/or the function TP) decide what kind (consistent, inconsistent, valid) of a formula is $U_j, j \in \{1,2,...,8\}$.

If $U_j, j \in \{1,2,...,8\}$ is consistent, find all its models.

1. $U_1 = (p \wedge q) \vee (\neg p \wedge \neg r) \rightarrow (q \leftrightarrow r)$;
2. $U_2 = (p \vee q \rightarrow r) \rightarrow (p \vee r \rightarrow q)$;
3. $U_3 = (p \wedge q \rightarrow r) \rightarrow (p \rightarrow r) \wedge q$;
4. $U_4 = (q \vee r \rightarrow p) \rightarrow (p \rightarrow r) \wedge q$;
5. $U_5 = (r \vee q) \vee (p \rightarrow \neg r) \rightarrow (p \leftrightarrow q)$;
6. $U_6 = (r \wedge q) \vee (\neg p \vee \neg r) \rightarrow (p \leftrightarrow q)$;
7. $U_7 = (q \wedge r \rightarrow p) \rightarrow (p \rightarrow r) \wedge q$;
8. $U_8 = (p \wedge r) \vee (\neg p \wedge \neg r) \rightarrow (q \leftrightarrow r)$.

Exercise 3.2.
Prove that the following formulas are tautologies using the semantic tableaux method:

1. permutation of the premises law: $(p \to (q \to r)) \to (q \to (p \to r))$;
2. separation of the premises law: $(p \wedge q \to r) \to (p \to (q \to r))$;
3. distribution of '$\to$' over '$\vee$': $(p \to q \vee r) \leftrightarrow (p \to q) \vee (p \to r)$;
4. distribution of '$\vee$' over '$\leftrightarrow$': $(p \vee (q \leftrightarrow r)) \leftrightarrow ((p \vee q) \leftrightarrow (p \vee r))$;
5. reunion of the premises law: $(p \to (q \to r)) \to (p \wedge q \to r)$;
6. distribution of implication: $(p \to (q \to r)) \leftrightarrow (p \to q) \to (p \to r)$;
7. distribution of '$\to$' over '$\wedge$': $(p \to q \wedge r) \leftrightarrow (p \to q) \wedge (p \to r)$;
8. distribution of '$\to$' over '$\leftrightarrow$': $(p \to (q \leftrightarrow r)) \leftrightarrow ((p \to q) \leftrightarrow (p \to r))$.

Exercise 3.3.
Using the semantic tableaux method, decide if the following logical consequences hold or not. If a logical consequence does not hold find an anti-model of it.

1. $p \to (\neg q \vee r \wedge s), p, \neg s \models \neg q$
2. $\neg p \to (\neg q \to r), r \vee q \models (\neg p \to q) \vee r$
3. $p \to (q \vee r \wedge s), p, \neg r \models q$
4. $p \to q, r \to t, p \wedge r \models q \wedge t$
5. $p \wedge (q \to r), q \vee r \models p \to (q \to r)$
6. $p \to q \models (r \to t) \to (p \wedge r \to q \wedge t)$
7. $p \wedge (q \to r), q \vee r \models p \to (q \to r)$
8. $p \to q \vee r \models (p \to q) \vee (p \to r)$

Exercise 3.4.
Write all the anti-models of the propositional formulas $U_1,...,U_8$ using the semantic tableaux method.

1. $U_1 = (p \vee q) \wedge \neg r \to p \wedge q \wedge r$;
2. $U_2 = q \wedge \neg p \wedge r \to \neg p \vee \neg(q \wedge r)$;
3. $U_3 = p \to (q \wedge r) \vee q \wedge \neg p$;
4. $U_4 = \neg p \vee (\neg q \vee r) \to q \vee \neg p \vee r$;
5. $U_5 = \neg p \vee (\neg q \vee \neg r) \to q \wedge \neg p$;
6. $U_6 = \neg p \vee (\neg q \wedge \neg r) \to q \wedge \neg p \wedge r$;
7. $U_7 = \neg p \vee \neg(q \wedge r) \to q \wedge \neg p$;
8. $U_8 = \neg(\neg p \vee q) \vee r \to \neg p \vee (\neg q \vee r)$.

Exercise 3.5.
Using the semantic tableaux method, prove the following properties in predicate logic:

1. '$\exists$' is semi-distributive over '$\wedge$':
   $\models (\exists x)(A(x) \wedge B(x)) \rightarrow (\exists x)A(x) \wedge (\exists x)B(x)$ and
   $\not\models (\exists x)A(x) \wedge (\exists x)B(x) \rightarrow (\exists x)(A(x) \wedge B(x))$
2. '$\forall$' is semi-distributive over '$\vee$':
   $\models (\forall x)A(x) \vee (\forall x)B(x) \rightarrow (\forall x)(A(x) \vee B(x))$ and
   $\not\models (\forall x)(A(x) \vee B(x)) \rightarrow (\forall x)A(x) \vee (\forall x)B(x)$
3. '$\exists$' is semi-distributive over '$\rightarrow$':
   $\models ((\exists x)A(x) \rightarrow (\exists x)B(x)) \rightarrow (\exists x)(A(x) \rightarrow B(x))$ and
   $\not\models (\exists x)(A(x) \rightarrow B(x)) \rightarrow ((\exists x)A(x) \rightarrow (\exists x)B(x))$
4. '$\forall$' is semi-distributive over '$\rightarrow$' :
   $\models (\forall x)(A(x) \rightarrow B(x)) \rightarrow ((\forall x)A(x) \rightarrow (\forall x)B(x))$ and
   $\not\models ((\forall x)A(x) \rightarrow (\forall x)B(x)) \rightarrow (\forall x)(A(x) \rightarrow B(x))$
5. $\models (\exists x)(A(x) \rightarrow B(x)) \rightarrow ((\forall x)A(x) \rightarrow (\exists x)B(x))$ and
   $\not\models ((\exists x)A(x) \rightarrow (\exists x)B(x)) \rightarrow (\forall x)(A(x) \rightarrow B(x))$
6. '$\exists$' is distributive over '$\vee$'
   $\models (\exists x)(A(x) \vee B(x)) \leftrightarrow (\exists x)A(x) \vee (\exists x)B(x)$
7. '$\forall$' is distributive over '$\wedge$'
   $\models (\forall x)(A(x) \wedge B(x)) \leftrightarrow (\forall x)A(x) \wedge (\forall x)B(x)$
8. $\vdash (\exists x)(P(x) \rightarrow Q(x)) \leftrightarrow ((\forall x)P(x) \rightarrow (\exists x)Q(x))$

Exercise 3.6.
Check the validity of the following first-order formulas using the semantic tableaux method:

1. $U_1 = (\forall x)(\forall y)P(x,y) \leftrightarrow (\exists x)(\forall y)P(x,y)$;
2. $U_2 = (\exists x)(\forall y)P(x,y) \leftrightarrow (\forall y)(\exists x)P(x,y)$;
3. $U_3 = (\forall y)(\exists x)P(x,y) \leftrightarrow (\exists y)(\exists x)P(x,y)$;
4. $U_4 = (\forall x)(\forall y)P(x,y) \leftrightarrow (\forall y)(\forall x)P(x,y)$;
5. $U_5 = (\forall y)(\forall x)P(x,y) \leftrightarrow (\forall x)(\exists y)P(x,y)$;
6. $U_6 = (\exists y)(\exists x)P(x,y) \leftrightarrow (\exists x)(\forall y)P(x,y)$;
7. $U_7 = (\exists y)(\exists x)P(x,y) \leftrightarrow (\forall x)(\exists y)P(x,y)$;
8. $U_8 = (\exists y)(\exists x)P(x,y) \leftrightarrow (\exists x)(\exists y)P(x,y)$.

Exercise 3.7.

Using the semantic tableaux method check whether the following logical consequences hold.

1. $(\forall x)(P(x) \to Q(x)), (\forall x)P(x) \models (\forall x)Q(x)$;
2. $(\forall x)(\forall y)(Q(x,y) \to P(x,y)), (\forall z)Q(z,z) \models (\forall x)P(x,x)$;
3. $P(a), (\forall x)(P(x) \to P(f(x))) \models (\forall x)P(x)$;
4. $(\exists x)(\forall y)(P(x,y) \to R(x)), (\forall x)(\forall y)P(x,y) \models (\exists z)R(z)$;
5. $(\forall x)(P(x) \to Q(x)), (\exists x)P(x) \models (\exists x)Q(x)$;
6. $(\forall x)(\exists y)(P(x,y) \to Q(x,y)), (\exists z)P(z,z) \models (\forall x)Q(x,x)$;
7. $(\exists x)(\forall y)(Q(x,y) \to P(x,y)), (\forall z)Q(z,z) \models (\exists x)P(x,x)$;
8. $(\forall x)(\forall y)(P(x,y) \to P(y,x)) \models (\forall x)P(x,x)$.

4. SEQUENT AND ANTI-SEQUENT CALCULI

In this chapter two complementary axiomatic systems: the *sequent* and *anti-sequent calculi*, for propositional and predicate logic are presented. They are used to check the validity/derivability and non-validity/non-derivability respectively in classical logics. We used as references the following papers: [1, 3, 5, 43, 45].

4.1. The sequent calculus

The sequent calculus axiomatic system, proposed by Gentzen in [22], is an improvement of the natural deduction system. It formalizes propositional/predicate logic and provides a direct and syntactic proof method used to check the validity and derivability of a propositional/predicate formula.

Definition 4.1. (syntax)
A ***sequent*** has the form: $X \Rightarrow Y$, where X and Y are finite sets of propositional/predicate formulas. X is called ***antecedent*** and Y is called ***consequent***.
In the sequent $U_1,U_2,\ldots,U_n \Rightarrow V_1,V_2,\ldots,V_m$

1. the symbol '$\Rightarrow$' has the meaning of 'provable ';
2. $U_1,U_2,\ldots,U_n$ are called hypotheses, assumptions;
3. $V_1,V_2,\ldots,V_m$ are formulas to be proved.

Definition 4.2.
A basic sequent has one of the forms:

1. $X,U \Rightarrow Y,U$; where the same formula, U, appears in both antecedent and consequent
2. $X \Rightarrow T$ (***true*** is provable from anything).
3. $F \Rightarrow Y$ (anything is provable from ***false***)

Definition 4.3. (semantics)
The sequent $U_1,U_2,\ldots,U_n \Rightarrow V_1,V_2,\ldots,V_m$ ***is true*** if and only if the formula $U_1 \wedge U_2 \wedge \ldots \wedge U_n \rightarrow V_1 \vee V_2 \vee \ldots \vee V_m$ is valid (tautology), meaning that from the conjunction of the hypotheses at least one of the formulas from the consequent can be proved.

Remarks:

- From the semantic point of view, the formulas from the antecedent are connected by conjunction and the formulas from the consequent are connected by disjunction.
- All basic sequents are true.

Definition 4.4.

Sequent calculus – an axiomatic (formal) system of predicate logic:

$Seq = (\Sigma_{Seq}, F_{Seq}, A_{Seq}, R_{Seq})$, where:

1. $\Sigma_{Seq} = \Sigma_{\Pr} \cup \{\Rightarrow\}$ is the vocabulary;
2. $F_{Seq} = \{X \Rightarrow Y \mid X, Y \subset F_{\Pr}\}$ is the set of all sequents;
3. A_{Seq} is the set of axioms and contains all the basic sequents;
4. $R_{Seq} = \{\neg_l, \neg_r, \wedge_l, \wedge_r, \vee_l, \vee_r, \rightarrow_l, \rightarrow_r, \forall_l, \forall_r, \exists_l, \exists_r\}$ is the set of inference rules.

Table 1. Sequent calculus – inference/reduction rules

connective/ quantifier	Introduction into antecedent (*left* - side rules)	Introduction into consequent (*right* - side rules)
$\neg$	$(\neg_l)\ \dfrac{X \Rightarrow Y, V}{X, \neg V \Rightarrow Y}$	$(\neg_r)\ \dfrac{U, X \Rightarrow Y}{X \Rightarrow Y, \neg U}$
$\wedge$	$(\wedge_l)\ \dfrac{X, U, V \Rightarrow Y}{X, U \wedge V \Rightarrow Y}$	$(\wedge_r)\ \dfrac{X \Rightarrow U, Y \quad X \Rightarrow V, Y}{X \Rightarrow U \wedge V, Y}$
$\vee$	$(\vee_l)\ \dfrac{X, U \Rightarrow Y \quad X, V \Rightarrow Y}{X, U \vee V \Rightarrow Y}$	$(\vee_r)\ \dfrac{X \Rightarrow U, V, Y}{X \Rightarrow U \vee V, Y}$
$\rightarrow$	$(\rightarrow_l)\ \dfrac{X \Rightarrow U, Y \quad X, V \Rightarrow Y}{X, U \rightarrow V \Rightarrow Y}$	$(\rightarrow_r)\ \dfrac{X, U \Rightarrow V, Y}{X \Rightarrow U \rightarrow V, Y}$
$\forall$	$(\forall_l)\ \dfrac{X, U[x \leftarrow t], (\forall x)U(x) \Rightarrow Y}{X, (\forall x)U(x) \Rightarrow Y}$	$(\forall_r)\ \dfrac{X \Rightarrow V(x), Y}{X \Rightarrow (\forall x)V(x), Y}$
$\exists$	$(\exists_l)\ \dfrac{X, U(x) \Rightarrow Y}{X, (\exists x)U(x) \Rightarrow Y}$	$(\exists_r)\ \dfrac{X \Rightarrow V[x \leftarrow t], (\exists x)V(x), Y}{X \Rightarrow (\exists x)V(x), Y}$

In the above rules, U, V are formulas and X, Y are finite sets of formulas, even empty sets, which remain unaffected by the application of rules. t is an appropriate term (variable, constant) used in existential and universal instantiations.

The rules have one or two ***premises*** (above the horizontal line) and a ***conclusion*** (below the horizontal line).

Dual rules:

- $(\wedge_l)$ and $(\vee_r)$;
- $(\vee_l)$ and; $(\wedge_r)$
- $(\forall_l)$ and $(\exists_r)$;
- $(\exists_l)$ and. $(\forall_r)$

If we apply the rules from the premises to the conclusion, they are called ***inference rules.*** The rules are called ***reduction rules*** if they are applied backwards, from the conclusion to the premises.

Note: Usually, the sequent calculus rules are applied as reduction rules.

Remarks:

- The application of the reduction rules $(\exists_l)$ and $(\forall_r)$ realizes the transformation of the corresponding quantified variable from a bound variable into a free one (the quantifier is eliminated).
- The result of applying $(\forall_l)$ and $(\exists_r)$ rules is the instantiation of the universal/existential formula with appropriate terms (constants, variables) and the duplication of the universal/existential formula for further instantiations.

Recommendations:

- use first the rules with one premise;
- use the rules $(\forall_r)$, $(\exists_l)$, which transform the bound variables into free ones, before the rules $(\forall_l)$, $(\exists_r)$ which instantiate the corresponding formulas with appropriate terms.

The ***sequent calculus method*** can be described as follows:

- We simplify (reduce) the initial sequent by successive applications of the reduction rules in order to obtain basic sequents.
- We build a binary tree (up-side down) representing the reduction process:
  - The root node (at the bottom) is labeled with the initial sequent. It is important to rename the bound variables such that they will be distinct in the initial sequent.
  - A sequent from a current node is reduced by applying the corresponding reduction rule that has it (the sequent) as conclusion. We add in the tree one or two successors of the current node and these are labeled with the premises of the applied rule.
  - The construction of the tree stops when all the leaf nodes are labeled with basic sequents (overlined) or with sequents that cannot be reduced any more.

Remarks:

- A branch having a leaf node labeled with a basic sequent is a closed branch and symbolizes validity.

- A branch having a leaf node labeled with a non-basic sequent that cannot be reduced any more is an open branch and symbolizes inconsistency.

Theorem 4.1. Soundness and completeness of sequent calculus
The sequent $U_1,U_2,\ldots,U_n \Rightarrow V_1,V_2,\ldots,V_m$ is true if and only if it can be reduced to basic sequents using reduction rules.

or

The sequent $U_1,U_2,\ldots,U_n \Rightarrow V_1,V_2,\ldots,V_m$ is true if and only if it can be derived from basic sequents using inference rules.

The following theoretic results provide a direct proof method used to solve the two decision problems (to check the validity / derivability) in classical logics.

Theorem 4.2.
Let V be a propositional/predicate formula.
$\vdash V$ (V is a theorem) *if and only if*
$\models V$ (V is a tautology) *if and only if*
the sequent $\Rightarrow V$ is true (it can be reduced to basic sequents).

Theorem 4.3.
Let $U_1,U_2,\ldots,U_n,V$ be propositional/predicate formulas.
$U_1,U_2,\ldots,U_n \vdash V$ *if and only if*
$U_1,U_2,\ldots,U_n \models V$ *if and only if*
the sequent $U_1,U_2,\ldots,U_n \Rightarrow V$ is true (it can be reduced to basic sequents).

From the above theorems we have that the following meta-symbols are equivalent:

- $\vdash$ (syntactic consequence, deductibility, derivability)
- $\models$ (logical consequence)
- $\Rightarrow$ (provability)

Propositional logic is decidable, therefore the following statements are true:

1. The binary tree is always finite.
2. If the initial sequent $\Rightarrow V$ is reduced to basic sequents, the propositional formula V is valid.
3. If the initial sequent $\Rightarrow V$ cannot be reduced to basic sequents, the propositional formula V is not valid. Its anti-models are provided by the non-basic sequents from the leaf nodes, assigning the truth value T to all the literals from the antecedent and the truth value F to all the literals from the consequent.

The ***semi-decidability of predicate logic*** is expressed as follows:

1. If the binary tree which represents the reduction process of the sequent $\Rightarrow V$ is finite, we can decide if the formula V is a tautology (all the leaf nodes contain basic sequents) or not (at least one leaf node contains a non-basic sequent).
2. If the binary tree which represents the reduction process of the sequent $\Rightarrow V$ is infinite, we cannot decide if the formula V is valid or not.

Example 4.1.

Prove that the syllogism rule: $V = (p \to q) \to ((q \to r) \to (p \to r))$ is a theorem using the sequent calculus method.

We build the up-side down binary tree corresponding to the reduction process of the initial sequent: $\Rightarrow (p \to q) \to ((q \to r) \to (p \to r))$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dots\dots\dots\dots}{\underline{p}, q \to r \Rightarrow r, \underline{p}} \quad \dfrac{\dots\dots\dots\dots}{\underline{p}, \underline{q} \Rightarrow r, \underline{q}} \quad \dfrac{\dots\dots\dots\dots}{p, q, \underline{r} \Rightarrow \underline{r}}}{p, q \to r \Rightarrow r, q \Rightarrow r}\,(\to_l)}{p, p \to q, q \to r \Rightarrow r}\,(\to_l)\text{ for } p \to q}{p \to q, q \to r \Rightarrow p \to r}\,(\to_r)}{p \to q \Rightarrow (q \to r) \to (p \to r)}\,(\to_r)}{\Rightarrow (p \to q) \to ((q \to r) \to (p \to r))}\,(\to_r)$$

All three leaf nodes contain basic sequents (the literals belonging to both antecedent and consequent are underlined), therefore the initial sequent was reduced to basic sequents. Thus, $\Rightarrow V$ is a true sequent and according to Theorem 4.2, *the formula* $V = (p \to q) \to ((q \to r) \to (p \to r))$ *is a theorem.*

Example 4.2.

Using the sequent calculus determine whether the following logical consequence: $\neg p \vee q, r \vee \neg q, \neg(p \wedge r) \models \neg p$ holds or not.

According to Theorem 4.3 we have to check if the initial sequent: $\neg p \vee q, r \vee \neg q, \neg(p \wedge r) \Rightarrow \neg p$ is true or not.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dots\dots\dots\dots}{\neg p \vee q, r \vee \neg q, \underline{p} \Rightarrow \underline{p}} \quad \dfrac{\dfrac{\dfrac{\dots\dots\dots\dots}{\neg p \vee q, \underline{r}, p \Rightarrow \underline{r}} \quad \dfrac{\dfrac{\dots\dots\dots\dots}{\neg q, \underline{p} \Rightarrow \underline{p}, r}\,(\neg_l)\text{ for } \neg p \quad \dfrac{\dots\dots\dots\dots}{\underline{q}, p \Rightarrow \underline{q}, r}\,(\neg_l)}{\neg p, \neg q, p \Rightarrow r} \quad \dfrac{\dots\dots\dots\dots}{q, \neg q, p \Rightarrow r}\,(\vee_l)}{\neg p \vee q, \neg q, p \Rightarrow r}\,(\vee_l)\text{ for } r \vee \neg q}{\neg p \vee q, r \vee \neg q, p \Rightarrow r}\,(\wedge_r)}{\neg p \vee q, r \vee \neg q, p \Rightarrow p \wedge r}\,(\neg_r)}{\neg p \vee q, r \vee \neg q \Rightarrow \neg p, p \wedge r}\,(\neg_r)}{\neg p \vee q, r \vee \neg q, \neg(p \wedge r) \Rightarrow \neg p}\,(\neg_l)$$

The above binary tree corresponding to the reduction of the initial sequent has four leaf nodes containing basic sequents (overlined). Therefore the initial sequent is true and *the logical consequence:* $\neg p \vee q, r \vee \neg q, \neg(p \wedge r) \models \neg p$ *holds.*

Example 4.3.

Prove that the formula $U = \neg(p \wedge q) \to \neg p \wedge \neg q$ is not valid.

The reduction of the initial sequent $\Rightarrow \neg(p \wedge q) \to \neg p \wedge \neg q$ is graphically symbolized as follows:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\underline{p \Rightarrow p}}{\Rightarrow p, \neg p}(\neg_r) \qquad \dfrac{q \Rightarrow p}{\Rightarrow p, \neg q}(\neg_r)}{\Rightarrow p, \neg p \wedge \neg q}(\wedge_r) \qquad \dfrac{\dfrac{p \Rightarrow q}{\Rightarrow q, \neg p}(\neg_r) \qquad \dfrac{\underline{q \Rightarrow q}}{\Rightarrow q, \neg q}(\neg_r)}{\Rightarrow q, \neg p \wedge \neg q}(\wedge_r)}{\Rightarrow p \wedge q, \neg p \wedge \neg q}(\wedge_r) \text{ for } p \wedge q}{\neg(p \wedge q) \Rightarrow \neg p \wedge \neg q}(\neg_l)}{\Rightarrow \neg(p \wedge q) \to \neg p \wedge \neg q}(\to_r)$$

The up-side down binary tree has two leaf nodes containing basic sequents ($p \Rightarrow p$ and $q \Rightarrow q$) and two leaf nodes with non-basic sequents ($q \Rightarrow p$ and $p \Rightarrow q$), therefore $\Rightarrow U$ is not a true sequent and U *is not a valid formula.*

The non-basic sequents provide the anti-models of the formula U assigning the truth value T to all the literals from the antecedent and F to all the literals from the consequent.

- the sequent $q \Rightarrow p$ provides the anti-model:

  $i_1 : \{p, q\} \to \{T, F\}$, $i_1(p) = F$, $i_1(q) = T$, so $i_1(U) = F$
- the sequent $p \Rightarrow q$ provides the anti-model:

  $i_2 : \{p, q\} \to \{T, F\}$, $i_2(p) = T$, $i_2(q) = F$, so $i_2(U) = F$

We conclude that the contingent formula U has two anti-models: i_1, i_2 and two models corresponding to the other two interpretations.

Example 4.4.

Determine if we can apply a property of distributivity on the left-hand side of the '$\to$' connective over the '$\wedge$' connective.

This property is expressed as: $(p \to q \wedge r) \equiv (p \to q) \wedge (p \to r)$ equivalent to:

$\models (p \to q \wedge r) \leftrightarrow (p \to q) \wedge (p \to r)$.

We replace '$\leftrightarrow$' by the conjunction of the '$\to$' in both directions.

$\models ((p \to q \wedge r) \to (p \to q) \wedge (p \to r)) \wedge ((p \to q) \wedge (p \to r) \to (p \to q \wedge r))$

and we have to check the validity of the formulas:

$U_1 = (p \to q \wedge r) \to (p \to q) \wedge (p \to r)$ and

$U_2 = (p \to q) \wedge (p \to r) \to (p \to q \wedge r)$.

The binary tree corresponding to the reduction of the initial sequent $\Rightarrow (p \to q \wedge r) \to (p \to q) \wedge (p \to r)$ for U_1 is depicted below:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dots\dots}{\underline{p}\Rightarrow q,\underline{p}}\quad\dfrac{\dots\dots}{p,\underline{q},r\Rightarrow\underline{q}}(\wedge_l)}{p,q\wedge r\Rightarrow q}(\rightarrow_l)}{p\rightarrow q\wedge r,p\Rightarrow q}(\rightarrow_r)}{p\rightarrow q\wedge r\Rightarrow p\rightarrow q}\quad\dfrac{\dfrac{\dfrac{\dfrac{\dots\dots}{\underline{p}\Rightarrow r,\underline{p}}\quad\dfrac{\dots\dots}{p,q,\underline{r}\Rightarrow\underline{r}}(\wedge_l)}{p,q\wedge r\Rightarrow r}(\rightarrow_l)}{p\rightarrow q\wedge r,p\Rightarrow r}(\rightarrow_r)}{p\rightarrow q\wedge r\Rightarrow p\rightarrow r}(\wedge_r)$$

$$\dfrac{p\rightarrow q\wedge r\Rightarrow(p\rightarrow q)\wedge(p\rightarrow r)}{\Rightarrow(p\rightarrow q\wedge r)\rightarrow(p\rightarrow q)\wedge(p\rightarrow r)}(\rightarrow_r)$$

The initial sequent was reduced to four basic sequents: $\dfrac{\dots\dots}{\underline{p}\Rightarrow q,\underline{p}}$; $\dfrac{\dots\dots}{p,\underline{q},r\Rightarrow\underline{q}}$; $\dfrac{\dots\dots}{\underline{p}\Rightarrow r,\underline{p}}$ and $\dfrac{\dots\dots}{p,q,\underline{r}\Rightarrow\underline{r}}$, therefore U_1 *is a valid formula.*

Similarly, we can prove that the formula U_2 is valid, therefore *the distributivity on the left of* '$\rightarrow$' *over* '$\wedge$' *is a valid law.*

Example 4.5.

Prove the validity of the predicate formula $U=(\forall x)P(x)\rightarrow P(a)\wedge P(b)$.

The reduction tree for the initial sequent $\Rightarrow U$ is the following:

$$\dfrac{\dfrac{\dfrac{\dots\dots}{(\forall x)P(x),\underline{P(a)}\Rightarrow\underline{P(a)}}(\forall_l)[x\leftarrow a]}{(\forall x)P(x)\Rightarrow P(a)}\quad\dfrac{\dfrac{\dots\dots}{(\forall x)P(x),\underline{P(b)}\Rightarrow\underline{P(b)}}(\forall_l)[x\leftarrow b]}{(\forall x)P(x)\Rightarrow P(b)}(\wedge_r)$$

$$\dfrac{(\forall x)P(x)\Rightarrow P(a)\wedge P(b)}{\Rightarrow(\forall x)P(x)\rightarrow P(a)\wedge P(b)}(\rightarrow_r)$$

The universal quantified formulas on both branches were instantiated with appropriate terms: the constant a and the constant b, respectively. The copies of these formulas are not needed further because basic sequents were obtained.

The initial sequent was reduced to two basic sequents, and thus U *is a tautology.*

Example 4.6.

Determine if the predicate formula $(\forall x)P(x)\vee(\forall x)Q(x)\rightarrow(\forall x)(P(x)\vee Q(x))$ is a tautology.

It is important to rename the bound variables such that they are distinct in the initial sequent: $\Rightarrow(\forall x)P(x)\vee(\forall y)Q(y)\rightarrow(\forall z)(P(z)\vee Q(z))$.

We build the up-side down binary tree corresponding to the reduction process.

- The application of the reduction rule $(\forall_r)$ realizes the transformation of z from a bound variable into a free one (the universal quantifier is eliminated).
- The result of applying the rule $(\forall_l)$ is the instantiation of the universal formula with appropriate terms (in this example x, y are replaced by z) and the duplication of the universal formula for further instantiations.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dots\dots\dots\dots\dots\dots\dots}{(\forall x)P(x),\underline{P(z)} \Rightarrow \underline{P(z)},Q(z)}(\forall_l),[x \leftarrow z] \quad \dfrac{\dots\dots\dots\dots\dots\dots\dots}{(\forall y)Q(y),\underline{\underline{Q(z)}} \Rightarrow P(z),\underline{\underline{Q(z)}}}(\forall_l),[y \leftarrow z]}{(\forall x)P(x) \Rightarrow P(z),Q(z) \quad (\forall y)Q(y) \Rightarrow P(z),Q(z)}(\vee_l)}{(\forall x)P(x) \vee (\forall y)Q(y) \Rightarrow P(z),Q(z)}(\vee_r)}{(\forall x)P(x) \vee (\forall y)Q(y) \Rightarrow P(z) \vee Q(z)}(\forall_r)}{(\forall x)P(x) \vee (\forall y)Q(y) \Rightarrow (\forall z)(P(z) \vee Q(z))}(\rightarrow_r)$$

$$\Rightarrow (\forall x)P(x) \vee (\forall y)Q(y) \rightarrow (\forall z)(P(z) \vee Q(z))$$

The reduction process is a finite one. Both leaf nodes of the reduction tree contain basic sequents, therefore the initial *formula is a tautology.*

Example 4.7.

Determine if there is a property of distributivity of the existential quantifier over conjunction.

The formula $U = (\exists x)P(x) \wedge (\exists x)Q(x) \leftrightarrow (\exists x)(P(x) \wedge Q(x))$ must be a tautology.

We prove that the formula $U = U_1 \wedge U_2$ is not valid, proving that $U_1 = (\exists x)P(x) \wedge (\exists x)Q(x) \rightarrow (\exists x)(P(x) \wedge Q(x))$ is not a tautology.

The reverse implication: $U_2 = (\exists x)(P(x) \wedge Q(x)) \rightarrow (\exists x)P(x) \wedge (\exists x)Q(x)$ is a tautology.

For building the initial sequents corresponding to U_1 and U_2 we have to rename the bound variables such that they are distinct in the initial sequents.

The reduction tree of the initial sequent

$\Rightarrow (\exists x)P(x) \wedge (\exists y)Q(y) \rightarrow (\exists z)(P(z) \wedge Q(z))$ corresponding to U_1 is as follows:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dots\dots\dots\dots\dots\dots\dots}{P(x),\underline{Q(y)} \Rightarrow Q(x),\underline{Q(y)},A \quad P(x),Q(y) \Rightarrow Q(x),P(y),A}(\wedge_r)}{P(x),Q(y) \Rightarrow Q(x),P(y) \wedge Q(y),(\exists z)(P(z) \wedge Q(z))}(\exists_r)[z \leftarrow y]}{\dots\dots\dots\dots\dots\dots\dots}}{\underline{\underline{P(x)},Q(y) \Rightarrow \underline{P(x)},A \quad P(x),Q(y) \Rightarrow Q(x),(\exists z)(P(z) \wedge Q(z))}(\wedge_r)}}{P(x),Q(y) \Rightarrow P(x) \wedge Q(x),(\exists z)(P(z) \wedge Q(z))}(\exists_r)[z \leftarrow x]}{P(x),Q(y) \Rightarrow (\exists z)(P(z) \wedge Q(z))}(\exists_l)}{(\exists x)P(x),(\exists y)Q(y) \Rightarrow (\exists z)(P(z) \wedge Q(z))}(\wedge_l)$$

$$\dfrac{(\exists x)P(x) \wedge (\exists y)Q(y) \Rightarrow (\exists z)(P(z) \wedge Q(z))}{\Rightarrow (\exists x)P(x) \wedge (\exists y)Q(y) \rightarrow (\exists z)(P(z) \wedge Q(z))}(\rightarrow_r)$$

We denote by A the existential formula $(\exists z)(P(z) \wedge Q(z))$.

In the first application of the rule $(\exists_r)$ for A we use the substitution $[z \leftarrow x]$, and in the second application the substitution $[z \leftarrow y]$ is used. The finite binary tree has two basic sequents (overlined) and one non-basic sequent.
The node which contains the sequent $P(x), Q(y) \Rightarrow Q(x), P(y), A$ is a leaf node because this sequent cannot be reduced any more (all the appropriate instantiations were applied).
The sequent $\Rightarrow U_1$ cannot be reduced to basic sequents, therefore U_1 *is not valid.*

Now we build the reduction tree of the initial sequent:
$\Rightarrow (\exists z)(P(z) \wedge Q(z)) \rightarrow (\exists x)P(x) \wedge (\exists y)Q(y)$, corresponding to the formula U_2:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dots\dots\dots}{P(z), Q(z) \Rightarrow \underline{P(z)}}\,(\exists_r),[x \leftarrow z] \quad \dfrac{\dots\dots\dots}{P(z), \underline{Q(z)} \Rightarrow \underline{Q(z)}}\,(\exists_r),[y \leftarrow z]}{P(z), Q(z) \Rightarrow (\exists x)P(x) \quad P(z), Q(z) \Rightarrow (\exists y)Q(y)}\,(\wedge_r)}{P(z), Q(z) \Rightarrow (\exists x)P(x) \wedge (\exists y)Q(y)}\,(\wedge_l)}{P(z) \wedge Q(z) \Rightarrow (\exists x)P(x) \wedge (\exists y)Q(y)}\,(\exists_l)}{(\exists z)(P(z) \wedge Q(z)) \Rightarrow (\exists x)P(x) \wedge (\exists y)Q(y)}\,(\rightarrow_r)$$

$$\Rightarrow (\exists z)(P(z) \wedge Q(z)) \rightarrow (\exists x)P(x) \wedge (\exists y)Q(y)$$

The sequent $\Rightarrow U_2$ was reduced to two basic sequents, therefore U_2 *is a valid formula.* As a conclusion, the formula U *is not a tautology* and thus *the property of distributivity of the existential quantifier over conjunction does not hold.*

Due to the fact that U_2 is a tautology, we have *the property of semi-distributivity of the existential quantifier over conjunction.*

Example 4.8.
Determine if the following predicate formula:
$U = (\exists x)(\forall y)(P(x,y) \rightarrow Q(x)) \rightarrow ((\forall x)(\forall y)P(x,y) \rightarrow (\exists s)Q(s))$ is valid.

The initial sequent, obtained after renaming the bound variables,
$\Rightarrow (\exists x)(\forall y)(P(x,y) \rightarrow Q(x)) \rightarrow ((\forall z)(\forall t)P(z,t) \rightarrow (\exists s)Q(s))$
is reduced according to the following binary tree.

We use the notations: $A = (\exists s)Q(s)$, $B(x) = (\forall y)(P(x,y) \rightarrow Q(x))$, $C = (\forall z)(\forall t)P(z,t)$, $D(x) = (\forall t)P(x,t)$ for the copies of the existential and universal formulas instantiated during the reduction process.
The reduction tree is a finite one and has two leaf nodes labeled with basic sequents (overlined), therefore the initial sequent is a true one and *the formula U is a tautology.*

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dots\dots\dots\dots}{\underline{P(x,x)},B(x),C,D(x)\Rightarrow \underline{P(x,x)},Q(x),A}\quad\dfrac{\dots\dots\dots\dots}{P(x,x),\underline{Q(x)},B(x),C,D(x)\Rightarrow \underline{Q(x)},A}\,(\rightarrow_l)}{P(x,x)\rightarrow Q(x),P(x,x),B(x),C,D(x)\Rightarrow Q(x),A}\,(\forall_l)[t\leftarrow x]}{P(x,x)\rightarrow Q(x),B(x),(\forall t)P(x,t),C\Rightarrow Q(x),A}\,(\forall_l)[y\leftarrow x,z\leftarrow x]}{(\forall y)(P(x,y)\rightarrow Q(x)),(\forall z)(\forall t)P(z,t)\Rightarrow Q(x),A}\,(\exists_r)[s\leftarrow x]}{(\forall y)(P(x,y)\rightarrow Q(x)),(\forall z)(\forall t)P(z,t)\Rightarrow (\exists s)Q(s)}\,(\exists_l)}{(\exists x)(\forall y)(P(x,y)\rightarrow Q(x)),(\forall z)(\forall t)P(z,t)\Rightarrow (\exists s)Q(s)}\,(\rightarrow_r)}{(\exists x)(\forall y)(P(x,y)\rightarrow Q(x))\Rightarrow (\forall z)(\forall t)P(z,t)\rightarrow (\exists s)Q(s)}\,(\rightarrow_r)$$

$$\Rightarrow (\exists x)(\forall y)(P(x,y)\rightarrow Q(x))\rightarrow((\forall z)(\forall t)P(z,t)\rightarrow(\exists s)Q(s))$$

4.2. The anti-sequent calculus

The anti-sequent calculus axiomatic system was introduced in paper [5] as a *complementary* system of sequent calculus and the corresponding proof method is used to check the non-validity and non-derivability in propositional/predicate logic. This method was then adapted for non-monotonic logics [36].

Definition 4.5. (syntax)
An ***anti-sequent*** has the form $X \not\Rightarrow Y$, where X (***antecedent***) and Y (***consequent***) are finite sets of propositional/predicate formulas. The meta-symbol $\not\Rightarrow$ means ***non-provable***.
An anti-sequent $X \not\Rightarrow Y$ is called a ***basic anti-sequent*** if all the formulas of X and Y are atomic formulas and $X \cap Y = \varnothing$.

Definition 4.6. (semantics)
Let $U_1,U_2,...,U_n,V_1,V_2,...,V_m$ be propositional/predicate formulas.
The ***anti-sequent*** $U_1,U_2,...,U_n \not\Rightarrow V_1,V_2,...,V_m$ ***is true*** if there is a model I of the set $\{U_1,U_2,...,U_n\}$ of formulas which is an anti-model of the set $\{V_1,V_2,...,V_m\}$.

Remark: The basic anti-sequents are true.

Definition 4.7. Anti-sequent calculus – axiomatic (formal) system
$Seq^c = (\Sigma^c_{Seq}, F^c_{Seq}, A^c_{Seq}, R^c_{Seq})$, where:

1. $\Sigma^c_{Seq} = \Sigma_{\Pr} \cup \{\not\Rightarrow\}$ is the vocabulary
2. $F^c_{Seq} = \{X \not\Rightarrow Y \mid X,Y \subset F_{\Pr}\}$ is the set of all anti-sequents
3. A^c_{Seq} is the set of axioms and contains all the basic anti-sequents.
4. $R^c_{Seq} = \{\neg^c_l, \neg^c_r, \wedge^c_l, \wedge^c_{r_1}, \wedge^c_{r_2}, \vee^c_{l_1}, \vee^c_{l_2}, \vee^c_r, \rightarrow^c_{l_1}, \rightarrow^c_{l_2}, \rightarrow^c_r, \exists^c_l, \exists^c_r\}$ is the set of inference/reduction rules

Table 2. Anti-sequent calculus – inference /reduction rules

Introduction into antecedent (*left* - side rules)	Introduction into consequent (*right* - side rules)
$(\neg_l^c)\ \dfrac{X \nRightarrow U,Y}{X,\neg U \nRightarrow Y}$	$(\neg_r^c)\ \dfrac{X,V \nRightarrow Y}{X \nRightarrow \neg V,Y}$
$(\wedge_l^c)\ \dfrac{X,U,V \nRightarrow Y}{X,U \wedge V \nRightarrow Y}$	$(\wedge_{r_1}^c)\ \dfrac{X \nRightarrow U,Y}{X \nRightarrow U \wedge V,Y}$ $(\wedge_{r_2}^c)\ \dfrac{X \nRightarrow V,Y}{X \nRightarrow U \wedge V,Y}$
$(\vee_{l_1}^c)\ \dfrac{X,U \nRightarrow Y}{X,U \vee V \nRightarrow Y}$ $(\vee_{l_2}^c)\ \dfrac{X,V \nRightarrow Y}{X,U \vee V \nRightarrow Y}$	$(\vee_r^c)\ \dfrac{X \nRightarrow U,V,Y}{X \nRightarrow U \vee V,Y}$
$(\rightarrow_{l_1}^c)\ \dfrac{X \nRightarrow U,Y}{X,U \rightarrow V \nRightarrow Y}$ $(\rightarrow_{l_2}^c)\ \dfrac{X,V \nRightarrow Y}{X,U \rightarrow V \nRightarrow Y}$	$(\rightarrow_r^c)\ \dfrac{X,U \nRightarrow V,Y}{X \nRightarrow U \rightarrow V,Y}$
$(\forall_l^c)\ \dfrac{\{X,U[x \leftarrow t] \nRightarrow Y \mid t-\text{term}\}}{X,(\forall x)U(x) \nRightarrow Y}$	$(\forall_r^c)\ \dfrac{X \nRightarrow V(x),Y}{X \nRightarrow (\forall x)V(x),Y}$
$(\exists_l^c)\ \dfrac{X,U(x) \Rightarrow Y}{X,(\exists x)U(x) \Rightarrow Y}$	$(\exists_r^c)\ \dfrac{\{X \Rightarrow V[x \leftarrow t],Y \mid t-\text{term}\}}{X \Rightarrow (\exists x)V(x),Y}$

In the reduction rules, U, V are formulas and X, Y are finite sets of formulas, even empty sets, which remain unaffected by the application of rules. t is an appropriate term (variable, constant) used in existential and universal instantiations.

The difference between *Table 1* and *Table 2* consists in splitting the rules with two premises from the sequent calculus into pairs of rules in the anti-sequent calculus. Thus, the exhaustive search in the sequent calculus becomes non-determinism in the anti-sequent calculus and the reduction process is a linear one.

Theorem 4.4. **(complementarity of sequent and anti-sequent calculi)**
The anti-sequent $U_1,U_2,\ldots,U_n \nRightarrow V_1,V_2,\ldots,V_m$ is true
if and only if
the sequent $U_1,U_2,\ldots,U_n \Rightarrow V_1,V_2,\ldots,V_m$ is not true.

Theorem 4.5.
A propositional/predicate formula V is not a theorem (not valid) *if and only if* the anti-sequent $\nRightarrow V$ is true.

According to Theorem 4.3, ***derivability is expressed in sequent calculus*** as follows:

$U_1 \wedge U_2 \wedge \ldots \wedge U_n \vdash V_1 \vee V_2 \vee \ldots \vee V_m$ *if and only if*

the sequent $U_1, U_2, \ldots, U_n \Rightarrow V_1, V_2, \ldots, V_m$ is true,

meaning that from the conjunction of the hypotheses at least one of the formulas from the consequent can be proved (derived).

Theorem 4.6.

The ***non-derivability*** is expressed ***in anti-sequent calculus*** as follows:

$U_1 \wedge U_2 \wedge \ldots \wedge U_n \nvdash V_1 \vee V_2 \vee \ldots \vee V_m$ *if and only if*

the anti-sequent $U_1, U_2, \ldots, U_n \not\Rightarrow V_1, V_2, \ldots, V_m$ is true,

meaning that from the conjunction of the hypotheses none of the formulas from consequent can be proved (derived).

The anti-sequent calculus proof method:

We simplify (reduce) the initial anti-sequent by successive applications of the reduction rules in order to obtain an axiom (basic anti-sequent). The conclusion about the validity/derivability is based on the following:

1. ***If the anti-sequent*** $\not\Rightarrow V$ ***is not true*** because it cannot be reduced to a basic anti-sequent, in all the possible ways of applying anti-sequent rules, ***then the formula*** V ***is valid***.

2. ***If the anti-sequent*** $\not\Rightarrow V$ ***is true*** because it was reduced to the basic anti-sequent $p_1, \ldots, p_k \not\Rightarrow q_1, \ldots, q_j$ (where $p_1, \ldots, p_k, q_1, \ldots, q_j$ are literals), ***then the formula*** V ***is not valid***.
   There is an interpretation, I, called anti-model, which evaluates the formula V as false, $I(V) = F$. A partial anti-model of V is obtained as follows: $I(p_1) = \ldots = I(p_k) = T$ and $I(q_1) = \ldots = I(q_j) = F$.

3. ***If the anti-sequent*** $U_1, U_2, \ldots, U_n \not\Rightarrow V_1, V_2, \ldots, V_m$ ***is not true*** because it cannot be reduced to a basic anti-sequent, in all the possible ways of applying anti-sequent rules, ***then the syntactic consequence:*** $U_1, U_2, \ldots, U_n \vdash V_1 \vee V_2 \vee \ldots \vee V_m$ ***holds.***

4. ***If the anti-sequent*** $U_1, U_2, \ldots, U_n \not\Rightarrow V_1, V_2, \ldots, V_m$ ***is true*** because it was reduced to the basic anti-sequent $p_1, \ldots, p_k \not\Rightarrow q_1, \ldots, q_j$ (where $p_1, \ldots, p_k, q_1, \ldots, q_j$ are literals), ***then none of the formulas*** $V_1, V_2, \ldots, V_m$ ***is derivable (provable) from the set*** $\{U_1, U_2, \ldots, U_n\}$ ***of hypotheses***.

There is an interpretation I, which is a model for the hypotheses, $I(\{U_1,U_2,...,U_n\})=T$ but it falsifies each of the formulas from the consequent: $I(V_1)=...=I(V_m)=F$.

A partial anti-model for the derivation $U_1 \wedge U_2 \wedge ... \wedge U_n \vdash V_1 \vee V_2 \vee ... \vee V_m$ is obtained as follows: $I(p_1)=...=I(p_k)=T$ and $I(q_1)=...=I(q_j)=F$.

Example 4.9.

Prove that the formula $U=\neg p \vee \neg q \to \neg(p \vee q)$ is not valid.

Using the **sequent calculus**:

The reduction tree of the initial sequent: $\Rightarrow \neg p \vee \neg q \to \neg(p \vee q)$ has two leaf nodes which contain basic sequents and two leaf nodes containing non-basic sequents. Therefore, the initial sequent is not true and thus *the formula U is not valid.*

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dots\dots}{p \Rightarrow p}}{p,\neg p \Rightarrow}(\neg_l) \quad \dfrac{p \Rightarrow q}{p,\neg q \Rightarrow}(\neg_l)}{p,\neg p \vee \neg q \Rightarrow}(\vee_l) \quad \dfrac{\dfrac{\dots\dots}{q \Rightarrow q}}{q,\neg q \Rightarrow}(\neg_l) \quad \dfrac{q \Rightarrow p}{q,\neg p \Rightarrow}(\neg_l)}{q,\neg p \vee \neg q \Rightarrow}(\vee_l)}{p \vee q,\neg p \vee \neg q \Rightarrow}(\vee_l) \text{ for } p \vee q}{\neg p \vee \neg q \Rightarrow \neg(p \vee q)}(\neg_r)}{\Rightarrow \neg p \vee \neg q \to \neg(p \vee q)}(\to_r)$$

The non-basic sequents $p \Rightarrow q$ and $q \Rightarrow p$ provide two anti-models of U:

$i_1 : \{p,q\} \to \{T,F\}, i_1(p)=T, i_1(q)=F, \quad i_1(U)=F$ and

$i_2 : \{p,q\} \to \{T,F\}, i_2(p)=F, i_2(q)=T, \quad i_2(U)=F$.

Using the **anti-sequent calculus**:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{q \not\Rightarrow p}{q,\neg p \not\Rightarrow}(\neg_l^c)}{q,\neg p \vee \neg q \not\Rightarrow}(\vee_{l_1}^c)}{p \vee q,\neg p \vee \neg q \not\Rightarrow}(\vee_{l_2}^c) \text{ for } p \vee q}{\neg p \vee \neg q \not\Rightarrow \neg(p \vee q)}(\neg_r^c)}{\not\Rightarrow \neg p \vee \neg q \to \neg(p \vee q)}(\to_r^c)$$

The initial anti-sequent $\not\Rightarrow \neg p \vee \neg q \to \neg(p \vee q)$ is true because it was reduced to the basic anti-sequent $q \not\Rightarrow p$, and thus *the formula U is not valid.*

Note the exhaustive search in sequent calculus and the linear reduction process in anti-sequent calculus.

From the basic anti-sequent $q \not\Rightarrow p$, the anti-model i_2 of U is obtained as follows:

$i_2 : \{p,q\} \to \{T,F\}, i_2(q)=T, i_2(p)=F$;

$i_2(U)=i_2(\neg p \vee \neg q \to \neg(p \vee q))=\neg F \vee \neg T \to \neg(F \vee T)=T \vee F \to \neg T = F$

Example 4.10.
Using the anti-sequent calculus determine if the following deduction holds:
$\neg p \vee q, r \vee \neg q, \neg(p \wedge r) \vdash p$.

An up-side down binary tree corresponding to the reduction process of the initial anti-sequent: $\neg p \vee q, r \vee \neg q, \neg(p \wedge r) \not\Rightarrow p$ is represented graphically as follows:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{q, r \not\Rightarrow p}{q, r \vee \neg q \not\Rightarrow p}(\vee_{l_1}^{c})}{\neg p \vee q, r \vee \neg q \not\Rightarrow p}(\vee_{l_2}^{c}) \text{ for } \neg p \vee q}{\neg p \vee q, r \vee \neg q \not\Rightarrow p, p \wedge r}(\wedge_{r_1}^{c})}{\neg p \vee q, r \vee \neg q, \neg(p \wedge r) \not\Rightarrow p}(\neg_{l}^{c})$$

The initial anti-sequent was reduced to a basic anti-sequent $q, r \not\Rightarrow p$, which provides an anti-model for the derivation(deduction). The truth value T is assigned to all the literals from the antecedent and F is assigned to all the literals from the consequent.
The anti-model $i: \{p, q, r\} \to \{T, F\}, i(p) = F, i(q) = T, i(r) = T$ satisfies the hypotheses of the deduction: $i(\neg p \vee q) = T$, $i(r \vee \neg q) = T$, $i(\neg(p \wedge r)) = T$ and falsifies the conclusion: $i(p) = F$.
We conclude that *the initial deduction does not hold*: $\neg p \vee q, r \vee \neg q, \neg(p \wedge r) \nvdash p$

Another version of the non-deterministic reduction process for the same initial anti-sequent does not end in a basic anti-sequent.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{q, r \not\Rightarrow p, r}{q, r \vee \neg q \not\Rightarrow p, r}(\vee_{l_1}^{c})}{\neg p \vee q, r \vee \neg q \not\Rightarrow p, r}(\vee_{l_2}^{c}) \text{ for } \neg p \vee q}{\neg p \vee q, r \vee \neg q \not\Rightarrow p, p \wedge r}(\wedge_{r_2}^{c})}{\neg p \vee q, r \vee \neg q, \neg(p \wedge r) \not\Rightarrow p}(\neg_{l}^{c})$$

Example 4.11.
Check the validity of the propositional formula $U = p \vee (q \wedge r) \to (p \vee q) \wedge (p \vee r)$ using the sequent and anti-sequent calculi.

Using the **sequent calculus:**
The initial sequent $\Rightarrow U$ is reduced to four basic sequents as we see in the reduction tree depicted below.
According to the soundness and completeness theorem, U *is a valid formula.*

$$\dfrac{\dfrac{\dfrac{\dfrac{\dots\dots}{\underline{p} \Rightarrow \underline{p},q}}{p \Rightarrow p \vee q}(\vee_r) \quad \dfrac{\dfrac{\dots\dots}{\underline{p} \Rightarrow \underline{p},r}}{p \Rightarrow p \vee r}(\vee_r)}{p \Rightarrow (p \vee q) \wedge (p \vee r)}(\wedge_r) \quad \dfrac{\dfrac{\dfrac{\dots\dots}{\underline{q},r \Rightarrow p,\underline{q}}}{q,r \Rightarrow p \vee q}(\vee_r) \quad \dfrac{\dfrac{\dots\dots}{q,\underline{r} \Rightarrow p,\underline{r}}}{q,r \Rightarrow p \vee r}(\vee_r)}{q,r \Rightarrow (p \vee q) \wedge (p \vee r)}(\wedge_r)}{q \wedge r \Rightarrow (p \vee q) \wedge (p \vee r)}(\wedge_l)$$

$$\dfrac{p \vee (q \wedge r) \Rightarrow (p \vee q) \wedge (p \vee r)}{\Rightarrow p \vee (q \wedge r) \rightarrow (p \vee q) \wedge (p \vee r)}(\vee_l)\ (\rightarrow_r)$$

Using the **anti-sequent calculus:**

There are four possible linear reduction processes for the initial anti-sequent $\not\Rightarrow U$, caused by four possible combinations of the application of ($\vee_{l_1}^c$ or $\vee_{l_2}^c$) and ($\wedge_{r_1}^c$ or $\wedge_{r_2}^c$) rules in the second and the third steps.

Process 1:

$$\dfrac{\dfrac{\dfrac{\dfrac{p \not\Rightarrow p,q}{p \not\Rightarrow p \vee q}(\vee_r^c)}{p \not\Rightarrow (p \vee q) \wedge (p \vee r)}(\wedge_{r_1}^c)}{p \vee (q \wedge r) \not\Rightarrow (p \vee q) \wedge (p \vee r)}(\vee_{l_1}^c)}{\not\Rightarrow p \vee (q \wedge r) \rightarrow (p \vee q) \wedge (p \vee r)}(\rightarrow_r^c)$$

Process 2:

$$\dfrac{\dfrac{\dfrac{\dfrac{p \not\Rightarrow p,r}{p \not\Rightarrow p \vee r}(\vee_r^c)}{p \not\Rightarrow (p \vee q) \wedge (p \vee r)}(\wedge_{r_2}^c)}{p \vee (q \wedge r) \not\Rightarrow (p \vee q) \wedge (p \vee r)}(\vee_{l_1}^c)}{\not\Rightarrow p \vee (q \wedge r) \rightarrow (p \vee q) \wedge (p \vee r)}(\rightarrow_r^c)$$

Process 3:

$$\dfrac{\dfrac{\dfrac{\dfrac{q,r \not\Rightarrow p,q}{q \wedge r \not\Rightarrow p \vee q}(\vee_r^c,\wedge_l^c)}{q \wedge r \not\Rightarrow (p \vee q) \wedge (p \vee r)}(\wedge_{r_1}^c)}{p \vee (q \wedge r) \not\Rightarrow (p \vee q) \wedge (p \vee r)}(\vee_{l_2}^c)}{\not\Rightarrow p \vee (q \wedge r) \rightarrow (p \vee q) \wedge (p \vee r)}(\rightarrow_r^c)$$

Process 4:

$$\dfrac{\dfrac{\dfrac{\dfrac{q,r \not\Rightarrow p,r}{q \wedge r \not\Rightarrow p \vee r}(\vee_r^c,\wedge_l^c)}{q \wedge r \not\Rightarrow (p \vee q) \wedge (p \vee r)}(\wedge_{r_2}^c)}{p \vee (q \wedge r) \not\Rightarrow (p \vee q) \wedge (p \vee r)}(\vee_{l_2}^c)}{\not\Rightarrow p \vee (q \wedge r) \rightarrow (p \vee q) \wedge (p \vee r)}(\rightarrow_r^c)$$

The anti-sequent $\not\Rightarrow U$ is not true because it was not reduced to a basic anti-sequent in none of the above four linear reduction processes, therefore U *is a valid formula.*

Example 4.12.
Apply the sequent calculus and anti-sequent calculus to prove the non-validity of the predicate formula:

$$U = (\forall x)(P(x) \vee Q(x)) \rightarrow (\forall x)P(x) \vee (\forall x)Q(x)\,.$$

For both methods we need to rename the bound variables such that they are distinct in the formula:

$$U \equiv (\forall x)(P(x) \vee Q(x)) \rightarrow (\forall y)P(y) \vee (\forall z)Q(z)\,.$$

Using the **sequent calculus**:

The initial sequent: $\Rightarrow (\forall x)(P(x) \vee Q(x)) \rightarrow (\forall y)P(y) \vee (\forall z)Q(z)$ is reduced according to the following up-side down binary tree.
We use the notation: $A = (\forall x)(P(x) \vee Q(x))\,.$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dots\dots\dots\dots}{Q(y),\underline{Q(z)},A \Rightarrow P(y),\underline{Q(z)}} \quad \dfrac{\dots\dots\dots\dots}{Q(y),P(z),A \Rightarrow P(y),Q(z)}(\vee_l)}{Q(y),P(z) \vee Q(z),A \Rightarrow P(y),Q(z)}(\forall_l)[x \leftarrow z]}{\underline{P(y)},A \Rightarrow \underline{P(y)},Q(z) \quad Q(y),(\forall x)(P(x) \vee Q(x)) \Rightarrow P(y),Q(z)}(\vee_l)}{P(y) \vee Q(y),A \Rightarrow P(y),Q(z)}(\forall_l)[x \leftarrow y]}{(\forall x)(P(x) \vee Q(x)) \Rightarrow P(y),Q(z)}(\forall_r)}{(\forall x)(P(x) \vee Q(x)) \Rightarrow (\forall y)P(y),(\forall z)Q(z)}(\vee_r)}{(\forall x)(P(x) \vee Q(x)) \Rightarrow (\forall y)P(y) \vee (\forall z)Q(z)}(\rightarrow_r)$$

$$\Rightarrow (\forall x)(P(x) \vee Q(x)) \rightarrow (\forall y)P(y) \vee (\forall z)Q(z)$$

There are two leaf nodes that contain basic sequents (overlined) and one leaf node containing the sequent $Q(y),P(z) \Rightarrow P(y),Q(z),A$. This sequent cannot be reduced any more (all the appropriate instantiations $([x \leftarrow y],[x \leftarrow z])$ for the universal formula $(\forall x)(P(x) \vee Q(x))$ were done).

The sequent $\Rightarrow U$ cannot be reduced to basic sequents, therefore U *is not a valid formula.*

Using the **anti-sequent calculus**:
The reduction tree of the initial anti-sequent:
$\not\Rightarrow (\forall x)(P(x) \vee Q(x)) \rightarrow (\forall y)P(y) \vee (\forall z)Q(z)$ is depicted below.

$$\dfrac{\dfrac{\dfrac{\dfrac{\{P(t)\vee Q(t) \not\Rightarrow P(y),Q(z) \mid t \text{ - term of the language}\}}{(\forall x)(P(x)\vee Q(x)) \not\Rightarrow P(y),Q(z)}(\forall_l^c)}{(\forall x)(P(x)\vee Q(x)) \not\Rightarrow (\forall y)P(y),(\forall y)Q(z)}(\forall_r^c)}{(\forall x)(P(x)\vee Q(x)) \not\Rightarrow (\forall y)P(y)\vee(\forall y)Q(z)}(\vee_r^c)}{\not\Rightarrow (\forall x)(P(x)\vee Q(x))\to(\forall y)P(y)\vee(\forall z)Q(z)}(\to_r^c)$$

There is an infinite number of premises and we have to prove that all of them can be reduced to basic anti-sequents.

The anti-sequent $P(t)\vee Q(t) \not\Rightarrow P(y),Q(z)$ is true if either $P(t) \not\Rightarrow P(y),Q(z)$ or $Q(t) \not\Rightarrow P(y),Q(z)$ are true anti-sequents.

If $t \neq y$, $S_1 : P(t) \not\Rightarrow P(y),Q(z)$ is a basic anti-sequent.

If $t = y$ the anti-sequent $Q(t) \not\Rightarrow P(y),Q(z)$ becomes:

$S_2 : Q(y) \not\Rightarrow P(y),Q(z)$, which is a basic anti-sequent.

We have proved that all the premises are true anti-sequents, therefore the initial anti-sequent ($\not\Rightarrow U$): $\not\Rightarrow (\forall x)(P(x)\vee Q(x))\to(\forall y)P(y)\vee(\forall z)Q(z)$ is true and thus *the formula U is not valid.*

From the basic anti-sequents S_1 and S_2 we can obtain an infinite number of anti-models of U. Such a generic anti-model is:

$I =< D,m >$ with

$\{y,z\}\subseteq D$ - the domain of interpretation, $|D|\geq 2$ and

$m(P): D\to\{T,F\}, m(Q): D\to\{T,F\}$

- $m(P)(y) = F$ (the atoms from the consequent of S_1 and S_2 must be evaluated as *false*)
- $m(P)(t) = T, \forall t \in D\setminus\{y\}$ (the atom from the antecedent of $S_1 : P(t), t \neq y$, must be evaluated as *true*)
- $m(Q)(z) = F$ (the atoms from the consequent of S_1 and S_2 must be evaluated as *false*)
- $m(Q)(y) = T$ (the atom from the antecedent of $S_2 : Q(y)$, must be evaluated as *true*)
- for $m(Q)(t), \forall t \in D\setminus\{y,z\}$, there is no restriction for the truth value assigned to it.

Note that an anti-model of U must have at least 2 distinct elements in the domain of interpretation.

Let us consider the interpretation $I_1 =< D_1, m_1 >$, where:

$D_1 = \{9,4\}$ is the domain of interpretation;

$m_1(P): \{9,4\} \to \{T,F\}, m_1(P)(x):"x \vdots 2"; \; m_1(P)(9) = F, m_1(P)(4) = T$

$m_1(Q): \{9,4\} \to \{T,F\}, m_1(Q)(x):"x \vdots 3"; \; m_1(Q)(9) = T, m_1(Q)(4) = F$

For evaluating the formula U under the interpretation I_1, with the finite domain $D_1 = \{9,4\}$, the universally quantified subformulas are replaced by the conjunction of their instances for $x = 4$ and $x = 9$.

$$v^{I_1}(U) = v^{I_1}((\forall x)(P(x) \vee Q(x))) \to v^{I_1}((\forall x)P(x) \vee (\forall x)Q(x)) =$$

$$= v^{I_1}((\forall x)(P(x) \vee Q(x))) \to v^{I_1}((\forall x)P(x)) \vee v^{I_1}((\forall x)Q(x)) =$$

$$= (4 \vdots 2 \vee 4 \vdots 3) \wedge (9 \vdots 2 \vee 9 \vdots 3) \to (4 \vdots 2 \wedge 9 \vdots 2) \vee (4 \vdots 3 \wedge 9 \vdots 3) =$$

$$= (T \vee F) \wedge (F \vee T) \to (T \wedge F) \vee (F \wedge T) = T \wedge T \to F \vee F = T \to F = F$$

I_1 evaluates the formula U as false, so I_1 *is an anti-model of* U *and thus* U *is not a valid formula.*

Conclusions:

- Both sequent and anti-sequent proof methods can be used to check the validity/derivability and non-validity/non-derivability in classical logics.
- It is more efficient to check validity/derivability using sequent calculus and non-validity/non-derivability using anti-sequent calculus.
- The basic anti-sequents or the non-basic sequents from the leaf nodes of the reduction tree provide anti-models of the initial formula or derivation (deduction).
- The semantic tableau method and the sequent calculus method are dual proof methods:

Semantic tableaux method	**Sequent calculus method**
- semantic and refutation method	- syntactic and direct method
- graphical representation: a binary tree	- graphical representation: an up-side down binary tree
- models are built	- anti-models are built
- a closed branch represents inconsistency	- a closed branch represents validity

4.3. Exercises

Exercise 4.1.

Using the sequent calculus method prove the validity of the following formulas:

1. the left-distributivity of ' $\rightarrow$ 'over ' $\wedge$ ':
   $U_1 = (p \rightarrow q \wedge r) \leftrightarrow (p \rightarrow q) \wedge (p \rightarrow r)$;
2. the "cut" law:
   $U_2 = (p \rightarrow q) \wedge (p \wedge q \rightarrow r) \rightarrow (p \rightarrow r)$;
3. the reunion and separation of the premises law:
   $U_3 = (p \rightarrow (q \rightarrow r)) \leftrightarrow (p \wedge q \rightarrow r)$;
4. the left-semi-distributivity of ' $\wedge$ ' over ' $\rightarrow$ ':
   $U_4 = p \wedge (q \rightarrow r) \rightarrow ((p \wedge q) \rightarrow (p \wedge r))$;
5. the permutation of the premises law:
   $U_5 = (p \rightarrow (q \rightarrow r)) \leftrightarrow (q \rightarrow (p \rightarrow r))$;
6. the left-distributivity of ' $\rightarrow$ ' over ' $\vee$ ':
   $U_6 = (p \rightarrow q \vee r) \leftrightarrow (p \rightarrow q) \vee (p \rightarrow r)$;
7. the left-distributivity of ' $\vee$ ' over ' $\rightarrow$ ':
   $U_7 = p \vee (q \rightarrow r) \leftrightarrow (p \vee q) \rightarrow (p \vee r)$;
8. the third axiom (A_3) of propositional logic:
   $U_8 = (p \rightarrow q) \leftrightarrow (\neg q \rightarrow \neg p)$.

Exercise 4.2.

Using the sequent/anti-sequent calculus method determine if the following relations hold or not.

1. $\neg p \rightarrow (\neg q \rightarrow r) \models (\neg p \rightarrow q) \vee r$;
2. $p \vee (q \rightarrow r), q \wedge r \models p \rightarrow (q \rightarrow r)$;
3. $p \wedge (q \rightarrow r), q \vee r \models p \rightarrow (q \rightarrow r)$;
4. $p \rightarrow r, q \rightarrow t, p \wedge q \models r \wedge t$;
5. $p \rightarrow (q \vee r \wedge s), p, \neg r \models q \vee r$;
6. $p \rightarrow (\neg q \vee r \wedge s), p, \neg s \models \neg q \vee s$;
7. $p \rightarrow q, r \rightarrow t, p \wedge r \models q \wedge t$;
8. $p \rightarrow (q \vee r \wedge s), p, \neg r \models q \wedge p$.

Exercise 4.3.

Prove the non-validity of the following formulas using the sequent/anti-sequent calculus. Find an anti-model for each of the following formulas:

1. $U_1 = (p \wedge q \rightarrow r) \wedge \neg(p \rightarrow (q \rightarrow r))$;
2. $U_2 = (p \rightarrow (q \rightarrow r)) \wedge \neg(p \rightarrow (q \rightarrow r))$;

3. $U_3 = (p \to (q \to r)) \wedge p \wedge q \wedge \neg r$;
4. $U_4 = (p \to r) \wedge \neg((\neg p \to q) \to (\neg q \to r))$;
5. $U_5 = (p \to (q \vee r)) \wedge \neg(p \to q) \wedge \neg(p \to r)$;
6. $U_6 = (p \to q) \wedge (p \wedge q \to r) \wedge (p \wedge \neg r)$;
7. $U_7 = (\neg p \vee q) \wedge \neg(\neg q \to \neg p)$;
8. $U_8 = (p \to (q \to r)) \wedge \neg((p \to q) \to (p \to r))$.

Exercise 4.4.
Using the anti-sequent calculus method prove that the following deductions do not hold. Find an anti-model for each of them.

1. $p \wedge r \vee \neg p \wedge \neg r \vdash q \leftrightarrow r$;
2. $q \wedge r \to p \vdash (p \to r) \wedge q$;
3. $r \wedge q \vee \neg p \vee \neg r \vdash p \leftrightarrow q$;
4. $r \vee q \vee (p \to \neg r) \vdash p \leftrightarrow q$;
5. $q \vee r \to p \vdash (p \to r) \wedge q$;
6. $p \wedge q \to r \vdash (p \to r) \wedge q$;
7. $p \vee q \to r \vdash (p \vee r) \to q$;
8. $p \wedge q \vee \neg p \wedge \neg r \vdash q \leftrightarrow r$.

Exercise 4.5.
Prove the following properties using the sequent calculus method.

1. ' $\forall$ ' is semi-distributive over ' $\vee$ ':
   $\models (\forall x)P(x) \vee (\forall x)Q(x) \to (\forall x)(P(x) \vee Q(x))$ and
   $\not\models (\forall x)(P(x) \vee Q(x)) \to (\forall x)P(x) \vee (\forall x)Q(x))$
2. ' $\forall$ ' is distributive over ' $\wedge$ '
   $\models (\forall x)(P(x) \wedge Q(x)) \leftrightarrow (\forall x)P(x) \wedge (\forall x)Q(x)$
3. ' $\exists$ ' is semi-distributive over ' $\wedge$ ':
   $\models (\exists x)(P(x) \wedge Q(x)) \to (\exists x)P(x) \wedge (\exists x)Q(x)$ and
   $\not\models (\exists x)P(x) \wedge (\exists x)Q(x) \to (\exists x)(P(x) \wedge Q(x))$
4. ' $\forall$ ' is semi-distributive over ' $\to$ ':
   $\models (\forall x)(P(x) \to Q(x)) \to ((\forall x)P(x) \to (\forall x)Q(x))$ and
   $\not\models ((\forall x)P(x) \to (\forall x)Q(x)) \to (\forall x)(P(x) \to Q(x))$
5. ' $\exists$ ' is semi-distributive over ' $\to$ ' :
   $\models ((\exists x)P(x) \to (\exists x)Q(x)) \to (\exists x)(P(x) \to Q(x))$ and
   $\not\models (\exists x)(P(x) \to Q(x)) \to ((\exists x)P(x) \to (\exists x)Q(x))$
6. ' $\forall$ ' is distributive over ' $\wedge$ ':
   $\models (\forall x)P(x) \wedge (\forall x)Q(x) \leftrightarrow (\forall x)(P(x) \wedge Q(x))$

7. $\models (\exists x)(P(x) \to Q(x)) \to ((\forall x)P(x) \to (\exists x)Q(x))$ and
   $\not\models ((\exists x)P(x) \to (\exists x)Q(x)) \to (\forall x)(P(x) \to Q(x))$
8. $\models (\exists x)(P(x) \wedge Q(x)) \to (\exists x)(P(x) \to Q(x))$
   $\not\models (\exists x)(P(x) \wedge Q(x)) \to (\forall x)P(x) \vee (\forall x)Q(x)$

Exercise 4.6.

Using the sequent/anti-sequent calculus methods determine if the following deductions hold or not.

1. $P(a), (\forall x)(P(x) \to P(f(x))) \vdash (\forall x)P(x)$;
2. $(\forall x)(P(x) \to Q(x)), (\forall x)P(x) \vdash (\forall x)Q(x)$;
3. $(\forall x)(\forall y)(Q(x,y) \to P(x,y)), (\forall z)Q(z,z) \vdash (\forall x)P(x,x)$;
4. $(\exists x)(\forall y)(P(x,y) \to R(x)), (\forall x)(\forall y)P(x,y) \vdash (\exists z)R(z)$;
5. $(\forall x)(P(x) \to Q(x)), (\exists x)P(x) \vdash (\exists x)Q(x)$;
6. $(\exists x)(\forall y)(Q(x,y) \to P(x,y)), (\forall z)Q(z,z) \vdash (\exists x)P(x,x)$;
7. $(\forall x)(\forall y)(P(x,y) \to R(x)), (\exists x)(\exists y)P(x,y) \vdash (\exists z)R(z)$;
8. $(\forall x)(\forall y)(P(x,y) \to P(y,x)) \vdash (\forall x)P(x,x)$.

5. RESOLUTION PROOF METHOD

In 1965 J.A. Robinson proposed resolution [50], an efficient and easily to be implemented proof method. The primary purpose of the resolution method is to check the consistency/inconsistency of a set of clauses. Resolution is used as a refutation method (proof by contradiction) to solve the decision problems of classical logics. This method was successfully adapted for non-classical logics (multivalent logics, modal logics, temporal logics and non-monotonic logics). The papers [1, 2, 12, 16, 18, 20, 21, 31, 32, 43, 45, 47, 52, 59, 60, 61, 62, 63] were used as bibliographic references.

5.1. Resolution method for propositional logic

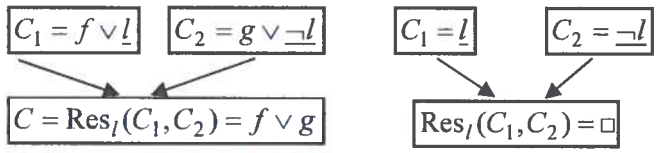
The resolution, as a syntactic proof method, is introduced by the ***formal (axiomatic) system:*** $\text{Res} = (\Sigma_{\text{Res}}, F_{\text{Res}}, A_{\text{Res}}, R_{\text{Res}})$ as was proposed in paper [62]:

- $\Sigma_{\text{Res}} = \Sigma_P - \{\rightarrow, \leftrightarrow, \wedge\}$ is the alphabet;
- $F_{\text{Res}} \cup \{\square\}$ is the set of well formed formulas;
  - F_{Res} is the set of all clauses built using the alphabet Σ_{Res};
  - $\square$ is the empty clause which does not contain any literal and it symbolizes inconsistency;
- $A_{\text{Res}} = \varnothing$ is the set of axioms ;
- $R_{\text{Res}} = \{res\}$ is the set of inference rules containing the ***resolution rule*** (***res***): $f \vee l,\ g \vee \neg l \vdash_{res} f \vee g$, where l is a literal and $f, g \in F_{\text{Res}}$.

Definition 5.1.
The clauses $C_1 = f \vee l$ and $C_2 = g \vee \neg l$ are called ***clashing clauses*** and they ***resolve upon the literal*** l. We use the notation: $C = \text{Res}_l(C_1, C_2) = f \vee g$, where C is called the ***resolvent*** of the ***parent clauses*** C_1 and C_2.

In the particular case: $C_1 = l$ and $C_2 = \neg l$, $\text{Res}_l(C_1, C_2) = \square$ (empty clause) which is inconsistent.

Graphical representation:



Resolution as an inference rule is a generalization of the rules: ***modus ponens***, ***modus tollens*** and the ***syllogism rule***.

Modus ponens rule: $p, p \to q \vdash_{mp} q$ can be written in the equivalent form:

$p,\ \neg p \vee q \vdash_{mp} q$, which is a particularization of the resolution rule.

Modus tollens rule (A_3 axiom):

$\vdash (p \to q) \to (\neg q \to \neg p)$, apply the reverse of the theorem of deduction

$p \to q \vdash \neg q \to \neg p$, apply the reverse of the theorem of deduction

$p \to q, \neg q \vdash \neg p$, the implication is written as a disjunction

$\neg p \vee q, \neg q \vdash \neg p$ ---- a particularization of the resolution rule.

Syllogism rule:

$\vdash (p \to q) \to ((q \to r) \to (p \to r))$, apply the reverse of the theorem of deduction

$p \to q \vdash (q \to r) \to (p \to r)$, apply the reverse of the theorem of deduction

$p \to q,\ q \to r \vdash p \to r$, the implications are written as disjunctions

$\neg p \vee q,\ \neg q \vee r \vdash \neg p \vee r$ ---- a particularization of the resolution rule

The resolution method consists of successive applications of the resolution rule upon an initial set of clauses, enriched in this process with new clauses (resolvents), in order to derive the empty clause. If the empty clause is derived, then the initial set of clauses is consistent, otherwise the initial set of clauses is consistent.

Algorithm propositional_resolution:

input: S – a set of propositional clauses

output: message "S is consistent" or "S is inconsistent"

begin

$S_0 := S$; $i := 0$;

do

choose two clashing clauses: $C_1, C_2 \in S_i$;

$C := \text{Res}(C_1, C_2)$; $S_{i+1} := S_i \cup \{C\}$;

if ($C = \square$) **then** write "S is inconsistent"; **exit;**

else $i := i + 1$;

end_if

until ($S_i = S_{i-1}$) // no new clauses can be derived

write "S is consistent";

end

Theorem 5.1. Soundness theorem [50]
If the empty clause is derived from the set S of propositional clauses using the resolution algorithm ($S \vdash_{Res} \square$), then S is an inconsistent set.

Theorem 5.2. Completeness theorem [50]
If the set S of propositional clauses is inconsistent, then the empty clause can be derived from S using the resolution algorithm ($S \vdash_{Res} \square$).

Theorem 5.3. Soundness and completeness theorem [50]
A set S of propositional clauses is inconsistent if and only if $S \vdash_{Res} \square$.

The resolution method is used as a refutation proof method (proof by contradiction) to solve the decision problems in propositional logic, according to the following theorems.

Theorem 5.4.
A propositional formula U is a tautology (theorem) if and only if the empty clause can be derived from the conjunctive normal form of $\neg U$, using the resolution algorithm.

U is a tautology (theorem) if and only if $\text{CNF}(\neg U) \vdash_{Res} \square$.

Theorem 5.5.
Let $U_1, U_2, ..., U_n, V$ be propositional formulas.
$U_1, U_2, ..., U_n \models V$ *if and only if* $U_1, U_2, ..., U_n \vdash V$ *if and only if*
$\text{CNF}(U_1 \wedge U_2 \wedge ... \wedge U_n \wedge \neg V) \vdash_{Res} \square$.

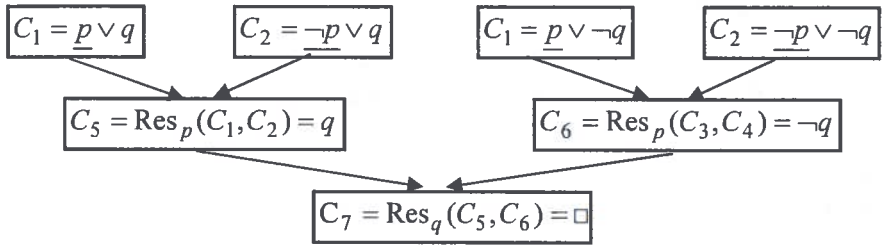
Example 5.1.
Using the resolution method prove that the following set of clauses is inconsistent.

$S = \{p \vee q,\ \neg p \vee q,\ p \vee \neg q,\ \neg p \vee \neg q\}$

We denote the propositional clauses as follows:

$C_1 = p \vee q$, $C_2 = \neg p \vee q$, $C_3 = p \vee \neg q$, $C_3 = \neg p \vee \neg q$.

The process of deriving the empty clause is symbolized by the binary tree depicted below:



$S \vdash_{Res} \square$ and according to the soundness theorem we conclude that *the set S of propositional clauses is inconsistent.*

The following theorem, inspired from Davis-Putman procedure [15], introduces a set of transformations used to simplify an initial set of clauses, preserving its consistency/inconsistency.

Theorem 5.6.
A set S of propositional clauses can be simplified, preserving its consistency/inconsistency, by applying the following transformations:

1. *Delete the clauses that are tautologies* (they contain at least one pair of opposite literals).
   example: $\neg p \vee \neg q \vee r \vee p$ is a tautology because $\neg p$, p are opposite literals
2. *Delete the clauses subsumed by other clauses of* S.
   C_1 ***is subsumed by*** C_2 or C_2 ***subsumes*** C_1 if there exists a clause C_3 such that $C_1 = C_2 \vee C_3$.
   example: $C_1 = t \vee \neg q \vee r \vee p$, $C_2 = r \vee p$,
   C_1 is subsumed by C_2, C_2 subsumes C_1 and $C_3 = t \vee \neg q$
   This transformation is based on the absorption property:
   $C_2 \wedge C_1 = C_2 \wedge (C_2 \vee C_3) \equiv C_2$,
   C_1 will be deleted from the set of clauses containing both C_1 and C_2.
3. *Delete every clause which contains a pure literal.*
   A ***pure literal*** is a literal that appears in a clause of S, but its negation does not appear in any clause of S.
   example: $S = \{C_1 = \neg p \vee \neg q \vee r,\ C_2 = r \vee p,\ C_3 = \neg p \vee \neg r\}$
   $\neg q$ is a pure literal in S and C_1 will be deleted.
4. *Let* $C = l$ *be a unit clause of* S. *Delete every clause containing* l *and delete* $\neg l$ *from every remaining clause.*
   example: $S = \{C_1 = \neg p \vee \neg q \vee r,\ C_2 = r \vee p,\ C_3 = \neg p \vee \neg r,\ C_4 = q\}$
   C_4 is a unit clause and is eliminated, the literal $\neg q$ is deleted from C_1,
   $S' = \{C_1' = \neg p \vee r,\ C_2 = r \vee p,\ C_3 = \neg p \vee \neg r\}$ is the simplified set.

Remarks:

- If $\varnothing$ is obtained after applying all the above transformations on a set S of clauses, then S is a consistent set.
- S is an inconsistent set of clauses if the empty clause ($\square$) is obtained after applying transformation 4.

Example 5.2.
Using the previous transformations, simplify the following set of propositional clauses: $S = \{C_1 = \neg p \vee p \vee r,\ C_2 = r \vee \neg r,\ C_3 = p \vee q,\ C_4 = p \vee q \vee r\}$.

- C_4 is subsumed by C_3 so C_4 will be eliminated.
- C_1 and C_2 are tautologies and they will be eliminated.
- only C_3 remains but it contains two pure literals: p and q and C_3 will be eliminated too.

After we applied the transformations, S becomes $S'=\varnothing$ and thus S *is consistent.*

Example 5.3.
Simplify the set $S=\{C_1=\neg q,\ C_2=q,\ C_3=p\vee q\vee r\}$ of propositional clauses.

- C_3 is subsumed by C_2 and it is eliminated.
- If we apply transformation 4, C_2 is a unit clause, C_2 is eliminated and C_1 becomes $C_1'=\square$ after the literal $\neg q$ is deleted.

After simplification, S becomes $S'=\{\square\}$ and we conclude that S *is an inconsistent set.*

5.2. Strategies for propositional resolution

The strategies used in the resolution process assure that all the possible clauses to be derived are generated and they also try to avoid the derivation of redundant and irrelevant clauses in order to obtain the empty clause.

1. The ***level saturation strategy*** [52] generates levels of resolvents corresponding to the exploration of the whole search space which contains all the possible resolvents:

- the initial level contains the initial set of clauses;
- we compute all the resolvents on a level (a resolvent has as one parent a clause from the level generated in the last iteration and the other parent belongs to one of the previous levels), we add them to the current level, and then we compute the next level.
- we continue until we obtain the empty clause or the last level is empty (no more resolvents can be derived).

Algorithm level-saturation-strategy:
input: S - the initial set of clauses;
output: message "S is inconsistent" or "S is consistent"
begin
// we generate the sequence $S^0,S^1,...,S^k$ containing sets of clauses (levels)
$S^0:=S$; $k:=0$; // initial level
do
{ $k:=k+1$;

$S^k := \{\text{Res}(C_1, C_2) \mid C_1 \in S^0 \cup S^1 \cup ... \cup S^{k-1}, C_2 \in S^{k-1}\}$;
// we eliminate the resolvents obtained in the current level but
// which appear already in the previous levels
$S^k := S^k \setminus (S^0 \cup S^1 \cup ... \cup S^{k-1})$;

} **until** ($\square \in S^k$ **or** $S^k = \varnothing$).

if ($\square \in S^k$) **then write** "S - inconsistent"

else //no more resolvents can be derived and $\square \notin S^k$

write "S - consistent" // no more resolvents can be derived

end_if

end

2. The ***deletion strategy*** [15]: the resolvents that are tautologies or are subsumed by other clauses in the set S of clauses are eliminated and they will not be used further in the resolution process because they produce redundant clauses.

Example 5.4.

Using the level saturation strategy combined with the deletion strategy check the consistency/inconsistency of the set $S = \{p \vee q,\ \neg p \vee q \vee \neg r,\ \neg q \vee r\}$ of clauses.

We generate the sequence $S^0, S^1, S^2, ...$ representing the levels of resolvents:

$S^0 = S = \{C_1 = p \vee q,\ C_2 = \neg p \vee q \vee \neg r,\ C_3 = \neg q \vee r\}$ - initial set of clauses

The first level of resolvents is generated:

$C_4 = \text{Res}_p(C_1, C_2) = q \vee \neg r$

$C_5 = \text{Res}_q(C_1, C_3) = p \vee r$

$C_6 = \text{Res}_q(C_2, C_3) = \neg p \vee \neg r \vee r$ - tautology

$C_7 = \text{Res}_r(C_2, C_3) = \neg p \vee q \vee \neg q$ - tautology

C_6 and C_7 will not be included in the first level.

$S^1 = \{C_4 = q \vee \neg r, C_5 = p \vee r\}$

The second level of resolvents is generated:

$C_8 = \text{Res}_r(C_4, C_5) = q \vee p = C_1$

$C_9 = \text{Res}_q(C_4, C_3) = r \vee \neg r$ - tautology

$C_{10} = \text{Res}_r(C_4, C_3) = q \vee \neg q$ - tautology

$C_{11} = \text{Res}_p(C_5, C_2) = q \vee r \vee \neg r$ - tautology

$C_{12} = \text{Res}_r(C_5, C_2) = \neg p \vee p \vee q$ - tautology

The clauses $C_9, C_{10}, C_{11}, C_{12}$, are tautologies, and C_8 belongs to the initial set of clauses as C_1, so they will not be introduced in the second level.

$S^2 = \varnothing$ with the meaning that no more resolvents can be generated.
The empty clause cannot be derived from S and thus S *is a consistent set.*

3. ***The set-of support strategy*** [65] avoids resolving two clauses belonging to a consistent subset of the initial set of clauses, because the resolvents derived from a consistent set are irrelevant in the process of deriving $\square$.

This strategy was inspired by the fact that usually, the set of premises (hypotheses, facts) of a deduction is consistent, and resolving two clauses from this set will not help us to derive the empty clause (inconsistency).

Definition 5.2. [62]
Let S be a set of clauses. A subset Y of S is called the ***support set of*** S, if the set $S \setminus Y$ is consistent. The ***set-of-support resolution*** is the resolution of two clauses that are not both from the set $S \setminus Y$. The resolvents generated during the resolution process are added to Y.

Example 5.5.
Using the set-of-support strategy, prove the following deduction:
$p \to (q \to r),\ r \wedge s \to t,\ u \to s \wedge \neg t \vdash p \wedge q \to \neg u$.
We denote the hypotheses and the conclusion of the deduction as follows:

$U_1 = p \to (q \to r)$,

$U_2 = r \wedge s \to t$,

$U_3 = u \to s \wedge \neg t$, $\qquad V = p \wedge q \to \neg u$

Applying the normalization algorithm we transform the above formulas into their conjunctive normal forms.

$\text{CNF}(U_1) = \neg p \vee \neg q \vee r$, $C_1 = \neg p \vee \neg q \vee r$

$\text{CNF}(U_2) = \neg r \vee \neg s \vee t$, $C_2 = \neg r \vee \neg s \vee t$

$\text{CNF}(U_3) = \neg u \vee (s \wedge \neg t) = (\neg u \vee s) \wedge (\neg u \vee \neg t)$, provides the clauses:

$C_3 = \neg u \vee s$ and $C_4 = \neg u \vee \neg t$

$\text{CNF}(\neg V) = p \wedge q \wedge r$ provides the clauses: $C_5 = p$, $C_6 = q$, $C_7 = u$

The set-of-support strategy is applied to the set of clauses:

$S = \{C_1, C_2, C_3, C_4, C_5, C_6, C_7\}$

We choose $Y = \{C_5, C_6, C_7\}$ as a support set of S, corresponding to $\neg V$ (the negation of the conclusion of the deduction). $S \setminus Y = \{C_1, C_2, C_3, C_4\}$ is a consistent set corresponding to the premises of the deduction.

This strategy avoids to resolve two clauses belonging to the consistent set $S \setminus Y$. C_1 and C_2 can resolve upon the literal r, but they belong to $S \setminus Y$, so in this strategy we will not resolve them.

Using the set-of-support strategy, we derive the empty clause from the set of clauses:

$S = \{ C_1 = \neg p \vee \neg q \vee r,\ C_2 = \neg r \vee \neg s \vee t,\ C_3 = \neg u \vee s,\ C_4 = \neg u \vee \neg t, C_5 = p,$
$C_6 = q,\ C_7 = u \}.$

The resolvents are as follows:

$C_8 = \text{Res}_p(C_1, C_5) = \neg q \vee r$

$C_9 = \text{Res}_q(C_8, C_6) = r$

$C_{10} = \text{Res}_r(C_9, C_2) = \neg s \vee t$

$C_{11} = \text{Res}_u(C_7, C_3) = s$

$C_{12} = \text{Res}_u(C_7, C_4) = \neg t$

$C_{13} = \text{Res}_s(C_{10}, C_{11}) = t$

$C_{14} = \text{Res}_s(C_{13}, C_{12}) = \square.$

We conclude that S is an inconsistent set and *the deduction holds*.

In order to make the resolution process more efficient, the **refinements of resolution** (***lock resolution, linear resolution, semantic resolution***) impose restrictions on the clashing clauses.

All the refinements and strategies of resolution preserve the soundness and the completeness properties. The question is: are these properties preserved when we combine refinements and strategies?

All the combinations of these refinements and strategies preserve the soundness property. The completeness is not preserved in some combinations.

Incompleteness: the initial set of clauses is inconsistent, but the empty clause cannot be derived because there are too many restrictions imposed by those strategies and refinements.

general resolution + deletion strategy is sound and complete

general resolution + set-of-support strategy is sound and complete

general resolution + deletion strategy + set-of-support strategy = sound and complete

5.3. Lock resolution

This refinement of resolution, introduced by R.S. Boyer in 1971 [10], is very efficient and easily to be implemented.

- Each occurrence of a literal from a set of clauses is arbitrarily indexed with an integer.
- ***Restriction***: the literals resolved upon must have the lowest indices in their clauses.
- The literals from resolvents inherit the indices from their parent clauses. If the parent clauses have a common literal, in the resolvent this literal will have the lowest index from the inherited indices.
- At the implementation level we must combine lock resolution with the level saturation strategy in order to check all the possible ways of deriving $\square$.

Theorem 5.7. **Soundness** [10]
Let S be a set of clauses having each literal *arbitrarily* indexed with an integer. If from S the empty clause can be derived using lock resolution ($S \vdash_{\text{Res}}^{lock} \square$), then S is inconsistent.

Theorem 5.8. **Completeness** [10]
Let S be a set of clauses having each literal *arbitrarily* indexed with an integer. If S is inconsistent, then there is a lock derivation of the empty clause from S ($S \vdash_{\text{Res}}^{lock} \square$).
The ***completeness*** property is not preserved if lock resolution is combined with the deletion strategy or with the set-of-support strategy.

The level saturation strategy is a safe method that can be used to prove that a set of clauses is consistent, since it assures the fact that all the possible clauses to be derived are obtained. This process can be shortened by using the combination of level saturation strategy with lock resolution.

Remarks:

1. In order to prove the consistency of a set S of clauses, lock resolution must be combined with the level saturation strategy:

   if $S^k = \varnothing$ (the last level of lock resolvents is empty)
   then the set ***S* is consistent**.

2. In order to prove inconsistency we do not need to use a strategy. It is enough to find a lock derivation of $\square$ to prove the inconsistency of a set S of clauses. If we combine lock resolution with the level saturation strategy the decision is as follows:

   if $\square \in S^k$ ($\square$ was derived as a lock resolvent in the level k)
   then the set ***S* is inconsistent**.

Example 5.6.

Check the consistency/inconsistency of the set S of propositional clauses.

$S = \{\neg r, p \vee \neg q, r \vee \neg p, \neg q\}$

a) The literals from the clauses are indexed as follows:

$C_1 =_{(1)} \neg r$, $C_2 =_{(3)} p \vee_{(2)} \neg q$, $C_3 =_{(4)} r \vee_{(5)} \neg p$, $C_4 =_{(6)} q$.

We use lock resolution + the level saturation strategy.

$S^0 = S$

The first level of resolvents is obtained as follows:

$S^1 = \{\text{Res}^{lock}(C_i, C_j) \mid C_i \in S^0, C_j \in S^0\}$

$C_5 = \text{Res}_r^{lock}(C_1, C_3) =_{(5)} \neg p$

$C_6 = \text{Res}_q^{lock}(C_2, C_4) =_{(3)} p$

$S^1 = \{C_5, C_6\}$

We generate the second level of resolvents:

$S^2 = \{\text{Res}^{lock}(C_i, C_j) \mid C_i \in S^1, C_j \in S^0 \cup S^1\}$

$C_7 = \text{Res}_p^{lock}(C_5, C_6) = \square$

$S^2 = \{C_7\}$

$\square \in S^2$, the empty clause was derived from S, therefore S *is an inconsistent set.*

b) We use another indexing of the literals from clauses:

$C_1 =_{(1)} \neg r$, $C_2 =_{(2)} p \vee_{(3)} \neg q$, $C_3 =_{(5)} r \vee_{(4)} \neg p$, $C_4 =_{(6)} q$

and we do not apply the level saturation strategy.

The empty clause is derived from S as follows:

$C_5 = \text{Res}_p^{lock}(C_2, C_3) =_{(3)} \neg q \vee_{(5)} r$

$C_6 = \text{Res}_q^{lock}(C_4, C_5) =_{(5)} r$

$C_7 = \text{Res}_r^{lock}(C_4, C_5) = \square$.

Example 5.7.

Using lock resolution and the level saturation strategy check the consistency/inconsistency of the following set of clauses:

$S = \{p \vee q, p \vee \neg q \vee \neg r, \neg p \vee r\}$.

The literals from the clauses are indexed as follows:

$C_1 =_{(1)} p \vee_{(2)} q$, $C_2 =_{(3)} p \vee_{(4)} \neg q \vee_{(5)} \neg r$, $C_3 =_{(7)} \neg p \vee_{(6)} r$,

$S^0 = S = \{C_1, C_2, C_3\}$ - the initial set of clauses

No lock resolvents can be generated ($S^1 = \varnothing$), the empty clause cannot be derived from S and we conclude that *the set S is consistent.*

Another indexing of literals of S will give the same result regarding consistency, but the resolution process will be longer, having more levels.
Let us consider the following indexing of the literals:

$C_1 =_{(1)} p \vee_{(2)} q$, $C_2 =_{(4)} p \vee_{(3)} \neg q \vee_{(5)} \neg r$, $C_3 =_{(6)} \neg p \vee_{(7)} r$,

$S^0 = S = \{C_1, C_2, C_3\}$ - the initial set of clauses

The first level of resolvents is generated: $S^1 = \{C_4\}$

$C_4 = \text{Res}_p(C_1, C_3) =_{(2)} q \vee_{(7)} r$

The second level of resolvents is generated: $S^2 = \{C_5\}$

$C_5 = \text{Res}_q(C_4, C_2) =_{(4)} p \vee_{(5)} \neg r \vee_{(7)} r \equiv T$ (contains a pair of opposite literals)

The third level of resolvents is generated: $S^3 = \{C_6\}$

$C_6 = \text{Res}_p(C_5, C_3) =_{(5)} \neg r \vee_{(7)} r \equiv T$

The fourth level of resolvents is the empty set, $S^4 = \varnothing$, since there is no clause where r is the literal with the lowest index in its clause.
The empty clause cannot be derived from S and thus S *is a consistent set.*

Example 5.8.
In this example we shall prove that *lock resolution + the deletion strategy is not complete.*
The set $S = \{p \vee q,\ \neg p \vee q,\ p \vee \neg q,\ \neg p \vee \neg q\}$ is inconsistent and we have the following *lock refutation* from S.
We index the literals of S as follows:

$C_1 =_{(1)} p \vee_{(2)} q$, $C_2 =_{(3)} \neg p \vee_{(4)} q$, $C_3 =_{(5)} p \vee_{(6)} \neg q$, $C_4 =_{(7)} \neg p \vee_{(8)} \neg q$

The lock resolvents are obtained:

$C_5 = \text{Res}_p^{lock}(C_1, C_2) =_{(2)} q$,

$C_6 = \text{Res}_p^{lock}(C_3, C_4) =_{(6)} \neg q$,

$C_7 = \text{Res}_p^{lock}(C_5, C_6) = \square$

Another indexing of the literals of S is used and we combine lock resolution with the deletion strategy:

$C_1 =_{(2)} p \vee_{(1)} q$, $C_2 =_{(3)} \neg p \vee_{(4)} q$, $C_3 =_{(5)} p \vee_{(6)} \neg q$, $C_4 =_{(8)} \neg p \vee_{(7)} \neg q$

Using the restriction imposed by lock resolution we can derive only the resolvents C_5 and C_6 which are tautologies: