VILNIUS UNIVERSITY

Julius Andrikonis

Mathematical Logic

Lecture Notes

Vilnius, 2012

Contents

Chapter 1

Introduction

1.1 A short history of logic

The term *logic* is derived from Ancient Greek language (gr. λογος, *logos* — reason, idea, word). A science of logic studies human thinking. More precisely it analyses different methods of thinking.

Example 1.1.1. Let's analyse two reasonings:

- If I have money, I will visit my parents. I haven't visited my parents. Therefore, I have no money.
- If I find my lecture notes, I will pass the exam. I haven't passed the exam. Therefore, I haven't found my lecture notes.

Both reasonings are different, but the way of thinking is the same: if p then q, not q, therefore not p.

Is such form of reasoning always correct? If so, then it is a *law of logic*. How to find other laws of logic? These (and many more) are the problems of logic.

However, natural language is usually too ambiguous to analyse. Thus thinking in logic is formalised. Words, reasonings, statements of natural language are presented using logical symbols. In other words, an *artificial language* is created. In high level of theoretical thinking it is nearly impossible to do without an artificial language. Different artificial languages are used to present mathematical formulas, equations of chemical reactions, etc.... An artificial language removes all the ambiguities of natural language, it provides optimal and the most precise way of presenting the results of research. However the definition of artificial language is similar to that of a natural language. Artificial language also has its alphabet and rules, which describe a way to make words. In logic words of language are usually called formulas. The variety of problems forces a variety of languages: propositional logic, predicate logic, modal logic, hybrid logic, etc....

Logic was developed as a branch of philosophy. In IV-VI centuries BC it already existed in Greece, China and India. The most renowned logician of those days is Greek philosopher Aristotle (BC 384–322), who created a logic theory, called *syllogistic*, which was a base for formal logic.

In syllogistic only reasonings of the form *if* F_1 *and* F_2*, then* F are analysed. Here F_1 and F_2 are called *hypothesis* and F is a *conclusion*. There must be exactly two hypothesis and one conclusion. Each statement (F_1, F_2 and F) can only be of such form:

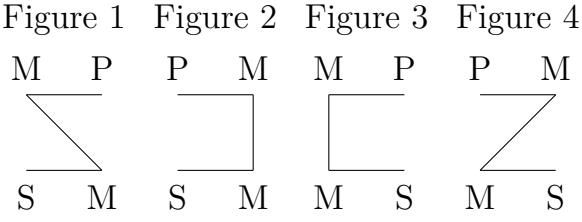
(a) Every S is P,

(i) Neither of S is P,

(e) Some of S are P,

(o) Some of S are not P.

Moreover, the three statements must contain three predicates, which are denoted P, M and S. Hypothesis F_1 must contain P and M, hypothesis F_2 — M and S and conclusion F must contain P and S. The order of predicates in both hypothesis can vary, however in the conclusion it is fixed: P is the first predicate and S is the second one. To better demonstrate the possible order of predicates in hypothesis, figures are used:



The reasoning consist of three statements, each can be either (a), (e), (i) or (o). Thus there are $4^3 = 64$ possible combinations: aaa, aai, aae, aao,.... These combinations are called *moduses* (lot. *modus* — method, rule, measure). Moreover, each of 64 moduses can represent any of the four figures, thus in total there are $64 \times 4 = 256$ possible combinations. All the combinations were analysed and as a result 19 sound reasonings (laws of logic) were found.

Another famous Greek logician is Zeno of Citium (BC 334–262), who lived in Athens and taught in a school called Stoa Poikile (gr. Ποικιλη στοα, Painted Porch). Because of the name of the school, his disciples were called *stoics*. They developed the basics of propositional logics. Stoics used logical operations of negation, conjunction, disjunction and implication to distinguish simple statements from composite ones. The notation of connectives wasn't used then. Reasonings were presented as natural language sentences with ordinal numbers in place of variables.

Example 1.1.2. A reasoning $\Big(\big((p \wedge q) \supset r\big) \wedge \big(p \wedge \neg r\big)\Big) \supset \neg q$ was denoted as follows:

If the 1st and the 2nd, then the 3rd.
Now the 1st holds but the 3rd does not.
Therefore, the 2nd does not hold.

Stoics founded the term *logic*. They created a deductive system, which contained five unquestionable or axiomatic reasonings and four inference rules, called *themata*. A reasoning was considered valid if it is either axiomatic or can be reduced to the axiomatic one by the rules.

The five axiomatic reasonings were:

1. If the 1st, then the 2nd. The 1st holds. Therefore the 2nd holds.
2. If the 1st, then the 2nd. The 2nd does not hold. Therefore the 1st does not hold.
3. It is not true that the 1st and the 2nd. The 1st holds. Therefore the 2nd does not hold.
4. Either 1st or the 2nd. The 1st holds. Therefore the 2nd does not hold.
5. Either 1st or the 2nd. The 1st does not hold. Therefore the 2nd holds.

Only two of the four themata are extant:

1. If from two statements a third one follows, then from the first one and the negation of the third one, the negation of the second one follows.
2. If from two statements a third one follows and from the third one and fourth one a fifth one follows, then from the first two statements and the fourth one the fifth one follows.

These ideas are still used in logic. E.g. axiomatic reasoning 1 is called *modus ponens*, axiomatic reasoning 2 is called *modus tolens*. Rules similar to themata 2 are called the cut rules.

After Aristotle and stoics there was a long period of stagnation, which lasted more than two thousand years.

It is still widely discussed who is the pioneer of modern mathematical logic. Different authors mention different names among which there is German mathematician G.W. Leibniz (1646–1716), English-born mathematician G. Boole (1815-1864), German mathematician F.L.G. Frege (1848–1925). But all of them were highly influenced by Aristotle.

British mathematician A. De Morgan (1806–1871) enriched logical laws by some properties of objects, analysed in algebra. G. Boole tried to establish logic as an exact science. His book *An Investigation of the Laws of Thought* was the first work, which analysed logical laws using mathematics. They were the first new ideas in logic after Aristotle and stoics. German mathematician E. Schröder (1841–1902) and Russian mathematician P.S. Poretsky (1846–1907) grounded propositional and predicate logic, by associating them with algebra.

During the centuries a lot of laws of logic were discovered. How? Usually a hypothesis was established and efforts were directed to deny it. I.e. an example was searched, which demonstrated that the hypothesis does not hold. If the efforts failed, a hypothesis was declared a law of logic. There were no proof in a mathematical sense. A set of laws of logic was first

systemised by German mathematician F.L.G. Frege. In 1879 he was the first one to develop a formal propositional calculus. He also showed that all the known and a lot of new statements are derivable in the calculus. For simplicity, the axioms will be presented using modern formal language.

1. $F \supset (G \supset F)$
2. $\Big((F \supset (G \supset H)\Big) \supset \Big((F \supset G) \supset (F \supset H)\Big)$
3. $\Big(F \supset (G \supset H)\Big) \supset \Big(G \supset (F \supset H)\Big)$
4. $(F \supset G) \supset (\neg G \supset \neg F)$
5. $\neg\neg F \supset F$
6. $F \supset \neg\neg F$

A calculus contains Modus ponens rule (if F and $F \supset G$, then G) and the rule of formula substitution. Later it was proved that axiom 3 is not necessary. It is derivable from other axioms.

It is worth mentioning that F.L.G. Frege defined the calculus before it was noticed that the validity of laws of logic can be checked using truth tables. Only six years later (in 1885) American mathematician and philosopher Ch.S. Peirce (1839–1914) developed the truth table method. The axiomatisation of F.L.G. Frege set an example for calculi that followed. After that, F.C.H.H. Brentano (1838–1917) defined calculus, which used only conjunction and negation, B.A.W. Russell (1872–1970) — a calculus with negation and disjunction. Another achievement of F.L.G. Frege in mathematising logic is introduction of *predicates*, *individual constants* and *quantifiers*. This provided a possibility to formalise mathematical expressions. G. Frege also started using symbol $\vdash$. A statement that F is valid was presented as $\vdash F$

The main results presented here are based on G. Frege calculus and the works of later logicians.

The main stimulus for mathematicians to take a closer look into formal logics was a discovery of antinomies (paradoxes). The biggest worry was caused by antinomy presented in the press by B.A.W. Russell in 1903.

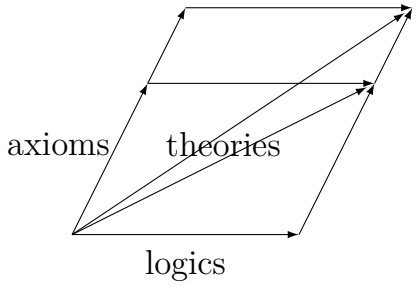
Definition 1.1.3 (Russell's Antinomy)**.** Suppose $\mathcal{M}$ is a set, which is composed of sets $\mathcal{A}_1, \mathcal{A}_2, \ldots$, that satisfy the condition: any set $\mathcal{A}_i$ is not an element of itself, i.e. set $\mathcal{A}_i$ is not an element of $\mathcal{A}_i$. It is not hard to reason that set $\mathcal{M}$ is an element of $\mathcal{M}$ iff it is not an element of $\mathcal{M}$.

Logic as an independent branch of mathematics with its own problems and methods was formed in the fourth decade of the XIX century. A serious impact to this was made by the works of Austrian logician K.F. Gödel (1906–1978). In 1930 he proved the completeness of predicate calculus. Therefore a predicate calculus became a system which was able to formalise mathematics. In contrast to other sciences, the main method to obtain new knowledge in mathematics is the proof and not the experiment. For example, by measuring the angles of many triangles it is possible to conclude that

the sum of inner angles of the triangle is 180°. However for mathematician this statement can be accepted as true only when it is proved by logical reasoning.

Now the proof of any theorem is actually only a sequence of formulas with some reasoning, which explains how a following formula can be obtained from the previous one(s). Formulas are understood uniformly, however the reasoning is often a cause of various ambiguities. Is it possible to find such rules of reasoning (logic), that can be expressed using mathematical formulas and used for proving theorems? If this would be possible, the proof of the theorem could be formulated as a sequence of formulas with names of the rules in between, that show which rule and already obtained formulas are used to derive the new formula. Having such sequence of reasoning it would be easy to check, if it is a proof of the theorem. The idea to create a universal language for all the mathematics and use it to formalise all the proofs was formulated by G.W. Leibniz. Such language would provide a tool to say which hypothesis is valid and which is not.

For a long time it was believed that there is only one logic, which is the same in everyday language as well as in mathematics, physics and other branches of science. The main aim of logicians were to define (discover) all the laws of logic. It was well known that by changing the system of axioms, the theory changes. For example, by changing the axiom of parallel lines to Lobachevsky axiom Lobachevsky non-Euclidian geometry is obtained. It seemed that the logic itself is absolute and stable. It was thought that mathematical theory could be presented as a diagonal of parallelogram. One side of the parallelogram would be an axiom system of the theory and the other one — logics (derivation rules):



However in 1921 Polish logician J. Łukasiewicz (1878–1956) described three-valued logic and a little bit later American logician E.L. Post (1897–1954) described m-valued ($m \geq 3$) logic. The illusion of the existence of a single logic collapsed. In 1930 Dutch mathematician A. Heyting (1898–1980) created *intuitionistic logic*, which can not be expressed using terms of classical logic. According to it, a new mathematical theory was described — *constructive mathematics*, in which Law of Excluded Middle[1] is not valid.

Modern computers provided many possibilities to apply logic. First of all, the logic for computer scientists is a formal language, which describes knowledge (information) about relational structures or models, queries, specifi-

[1]Tertium non datur — a statement is true or it is not true: $p \vee \neg p$.

cations. One of the task in informatics is to create new logic, which could be applied to solve new problems. The following logics were created for the needs of computer science:

- dynamic logic (in 1969 by C.A.R. Hoare (born 1934)),
- temporal logic (in 1977 by A. Pnueli (1941–2009)),
- descriptive logic (in 1984 by M. Schmidt-Schauss and G. Smolka),
- hybrid logic (in 1985 m. by S. Passy (born 1956) and T. Tinchev).

The creation of these new non-classical logics were inspired by *modal logic*. For a long time it was part of philosophy and known as necessity and possibility logic. After discovering that it is convenient for defining relations and analysing computer programs, its role changed. Soon the works about applying modal logics in informatics, artificial intelligence and linguistics appeared. Using modal logics, new formalisms are introduced, which enable analysis of time, knowledge, belief, etc…. First systems of modal logic were created by American logician C.I. Lewis (1882–1964) in 1918 and 1932 and called *S1*, *S2*, *S3*, *S4* and *S5*. Two of them — *S4* and *S5* — became famous in modal logic theory. They are still analysed and widely applied.

Different systems, which are analogous to *S4* was defined by Russian logician I. Orlov (1886–1936) and Austrian logician K.F. Gödel, approximately at the same time. They interpreted the necessity operator in a different way — as the operator of provability. A system, similar to *S5*, was already defined in 1906 by Scottish mathematician H. MacColl (1837–1909).

The first textbook *Principia Mathematica*, dedicated to mathematical logic and its applications, was published in 1910. The authors were B.A.W. Russell and A.N. Whitehead (1861–1947). The book contains such sentence:

> The fact that all Mathematics is Symbolic Logic is one of the greatest discoveries of our age.

Later during several years two more volumes were published. With the appearance of the book, new stage of logic development started. The next textbook appeared only in 1928. It was *Grundzüge der theoretischen Logik* by D. Hilbert (1862–1943) and W.F. Ackermann (1896-1962). Later *Grundlagen der Mathematik* by D. Hilbert and P.I. Bernays (1888–1977) was published (first volume in 1934 and the second volume in 1939). The later book completed the formation of logic as a independent branch of mathematics. A lot of symbols presented in these books are still used today.

Logical operation	B.A.W. Russell A.N. Whitehead	D. Hilbert P.I. Bernays
negation	$\sim p$	$\bar{p}$
conjunction	$p \,.\, q$	$p \,\&\, q$
disjunction	$p \vee q$	$p \vee q$
implication	$p \supset q$	$p \to q$
equivalence	$p \equiv q$	$p \sim q$

Symbols $\sim$, $\vee$, $\equiv$, $\supset$ were created in 1908 by B.A.W. Russell, . (to denote conjunction) — G. Boole. The author of $\neg$, $\wedge$ is A. Heyting (1929). The symbol & was created in 1924 by M.I. Schönfinkel (1889–1942), $\to$ — in 1917 by D. Hilbert, and $\leftrightarrow$ in 1940 by A. Tarski (1901–1983).

Now, when the computers are widely used, logic is becoming the branch of computer science. Between 1955 and 1956 (i.e. 10 years after the first computer was produced) Americans A. Newell (1927–1992), H.A. Simon (1916–2001) and J.C. Shaw (1922–1991) created a program *Logic Theorist*, which proved 38 out of 52 theorems of propositional logic, which were provided in three-volume textbook *Principia Mathematica* (1910-1913). A public demonstration of the program occurred on 9$^{\text{th}}$ August, 1956. It was the first program with *artificial intelligence*. The concept of artificial intelligence, meaning an artificial ability to think logically, was established in 1955 by American computer scientist J. McCarthy (1927-2011), who founded the first artificial intelligence laboratory together with M.L. Minsky (born 1927). He is the author of LISP programming language. Although similar program was created earlier by M. Davis (born 1928), however published results with the description of the program appeared later than about Logic Theorist. Both programs used some specific methods, which were suitable only to solve the analysed problems and because of that they didn't have more influence on later works about automatic theorem proving. The initiator of automatic theorem proving is Chinese-American logician H. Wang (1921–1995). Using IBM computer in 1959 he proved approximately 400 laws of propositional logic and first-order logic with equality, which were presented in three-volume textbook *Principia Mathematica*.

In 1965 American logician J.A. Robinson (born 1928) described a new derivation search method for predicate logic formulas, called *the resolution calculus*. Soon a new branch of informatics were formed, called *logic programming* (program — a finite sequence of logic formulas), which was based on the tactics of derivation search using resolution calculus. The initiators of logic programming are J.M. Foster and E.W. Elcock with their system ABSYS. In logic programming the input data and the query (aim) are defined using logic formulas. The result of the query is obtained by deduction using the input data. The most popular logic programming language is *PROLOG* (PROgramming in LOGic). It was created in 1971 by French mathematician A. Colmerauer (born 1941). A program in PROLOG is a sequence of first-order logic formulas, that satisfy certain conditions. The search of the result of the query is carried out using resolution method and linear tactics. In 1985 and 1986 French logician L. Farinas presented a

method to extend PROLOG by incorporating formulas of modal logic and called the new programming language MOLOG.

1.2 Important notation

In general, notation is introduced together with the concept it describes. Only some basic notation is provided here.

First of all, sets of numbers are denoted as follows:

- Natural numbers — $\mathbb{N}$; it is assumed that $0 \in \mathbb{N}$;
- Integer numbers — $\mathbb{Z}$;
- Rational numbers — $\mathbb{Q}$;
- Real numbers — $\mathbb{R}$;

Functions are defined in a following way: $f : \mathcal{A} \to \mathcal{B}$ meaning that function f is defined in set $\mathcal{A}$ and returns values in set $\mathcal{B}$. E.g. function $f : \mathbb{N} \times \mathbb{N} \to \mathbb{Q}$ has two natural arguments and the result is a rational number. Sign $\times$ here denotes Cartesian product.

Intervals of numbers are denoted as follows $[a, b]$. This notation means the interval between a and b, $a, b \in [a, b]$. I.e. $[a, b] = \{c : a \leqslant c \leqslant b\}$. If some edge does not belong to some interval, then respective square bracket is replaced by a parenthesis: $b \in (a, b]$, but $a \notin (a, b]$.

Expression of the form $\{{}^{y_1}/_{x_1}, {}^{y_2}/_{x_2}, \ldots, {}^{y_n}/_{x_n}\}$ is called a *substitution*. They are denoted by small Greek letters: α, β, γ. If $\alpha = \{{}^{y_1}/_{x_1}, {}^{y_2}/_{x_2}, \ldots, {}^{y_n}/_{x_n}\}$, then notation $F\alpha$ represents an expression, which is obtained from F by replacing all the occurrences of x_i by y_i for every $i \in [1, n]$. The type of expression F as well as of expressions y_i is not important, x_i are usually variables. Let α and β be two substitutions:

$$\alpha = \{{}^{y'_1}/_{x'_1}, {}^{y'_2}/_{x'_2}, \ldots, {}^{y'_n}/_{x'_n}\}$$

$$\beta = \{{}^{y''_1}/_{x'_{i_1}}, {}^{y''_2}/_{x'_{i_2}}, \ldots, {}^{y''_k}/_{x'_{i_k}}, {}^{z''_1}/_{x''_1}, {}^{z''_2}/_{x''_2}, \ldots, {}^{z''_m}/_{x''_m}\}$$

Here $x''_i \neq x'_j$ for any $i \in [1, m]$ and $j \in [1, n]$. A composition of α and β is a substitution:

$$\alpha \circ \beta = \{{}^{y'_1\beta}/_{x'_1}, {}^{y'_2\beta}/_{x'_2}, \ldots, {}^{y'_n\beta}/_{x'_n}, {}^{z''_1}/_{x''_1}, {}^{z''_2}/_{x''_2}, \ldots, {}^{z''_m}/_{x''_m}\}$$

In fact if $\gamma = \alpha \circ \beta$, then $F\gamma = (F\alpha)\beta$.

The word *iff* is used to denote the phrase *if and only if*.

1.3 Propositional logic

1.3.1 Syntax and semantics

As it can be implied from the name, propositional logic analyses *propositions*. Here propositional logic is denoted as PC.

Definition 1.3.1. A *proposition* is a sentence, which is either true or false.

Example 1.3.2. The following sentences are propositions:

- The sun is shining.
- I have never been to Australia.
- If I study hard, I will pass algebra and logic exams.

The following sentences are not propositions:

- Is it raining?
- Oh, what a beautiful day!
- Could you please pass me some salt?

As it can be seen from the example above, some propositions can be broken down into simpler ones. E.g. statement *If I study hard, I will pass algebra and logic exams* can be split into three simpler propositions: *I study hard*, *I will pass algebra exam* and *I will pas logic exam*. Such proposition is *composite*. However it is usually defined that composite statements are formed from the simpler ones using *logical operators*. Here these operators are presented:

- Negation. Denoted $\neg$. It is read as *Not something*. Here instead of *something* any proposition can be inserted.
- Conjunction. Denoted $\wedge$. It is read as *something and something*.
- Disjunction. Denoted $\vee$. It is read as *something or something*.
- Implication. Denoted $\supset$. It is read as *if something, then something*.
- Equivalence. Denoted $\leftrightarrow$. It is read as *something is equivalent to something*.

To formalise the statements, propositions are replaced by *propositional variables*. Here propositional variables are denoted by small Latin letters with or without indexes: p, q, r, p_1, p_2, q_1,.... Using propositional variables and (or) logical operators composite propositions, called *formulas*, are produced.

Definition 1.3.3. *Propositional formula* is defined recursively as follows:

- Propositional variable is a propositional formula. It is also called *atomic propositional formula*.
- If F is a propositional formula, then $(\neg F)$ is a propositional formula too.
- If F and G are propositional formulas, then $(F \wedge G)$, $(F \vee G)$, $(F \supset G)$, $(F \leftrightarrow G)$ are propositional formulas too.

Here formulas are denoted by capital Latin letters ($F, G, H, F_1, \dots$). Outermost parentheses of formulas are always omitted, as well as some inner parentheses if the order of application of logical operators is clear. The priorities of logical operators are $\neg$, $\wedge$, $\vee$, $\supset$, $\leftrightarrow$, where $\neg$ has the highest priority and $\leftrightarrow$ — the lowest.

Example 1.3.4. Formula $\Big((p \wedge q) \vee (\neg p)\Big)$ can be presented as $p \wedge q \vee \neg q$. The priority of $\neg$ is highest and the priority or $\wedge$ is higher than that of $\vee$, thus parentheses do not add any additional information.

However, formula $\Big((p \vee q) \wedge p\Big)$ can not be written as $p \vee q \wedge p$, because $\wedge$ has higher priority than $\vee$ and the latter formula is in fact $\Big(p \vee (q \wedge p)\Big)$.

Every proposition can be true (denoted $\top$) or false (denoted $\bot$). This is true for composite propositions as well. Moreover, to find if a composite proposition is true, it must be checked if the simpler ones, are true. The truth of the formula is defined by *interpretation*:

Definition 1.3.5. An *interpretation* of propositional formula F is function $\nu : \mathcal{P} \to \{\top, \bot\}$, where $\mathcal{P}$ is a set of all the propositional variables of F.

An interpretation only defines the value of every propositional variable. A value of the formula, which consists of propositional variables, is defined recursively according to logical operators of the formula. The fact that formula F is true with interpretation ν is denoted $\nu \vDash F$. The fact that formula F is false with interpretation ν is denoted $\nu \nvDash F$.

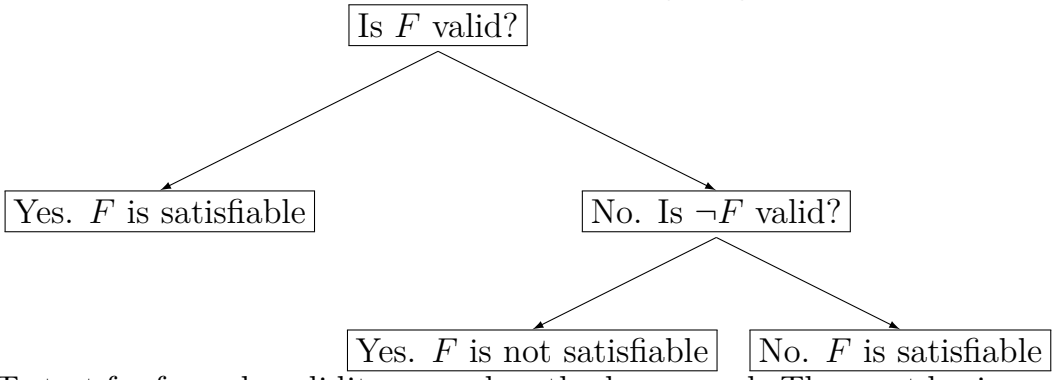
Definition 1.3.6. Let's say that ν is an interpretation of propositional formula F, then:

- if F is a propositional variable, then $\nu \vDash F$ iff $\nu(F) = \top$.
- if $F = \neg G$, then $\nu \vDash F$ iff $\nu \nvDash G$.
- if $F = G \wedge H$, then $\nu \vDash F$ iff $\nu \vDash G$ and $\nu \vDash H$.
- if $F = G \vee H$, then $\nu \nvDash F$ iff $\nu \nvDash G$ and $\nu \nvDash H$.
- if $F = G \supset H$, then $\nu \nvDash F$ iff $\nu \vDash G$ and $\nu \nvDash H$.
- if $F = G \leftrightarrow H$, then $\nu \vDash F$ iff $\nu \vDash G$ and $\nu \vDash H$ or $\nu \nvDash G$ and $\nu \nvDash H$.

If there is an interpretation with which F is true, then it is said that F is *satisfiable*. If F is true with every possible interpretation of F, then it is said that F is *valid* and denoted $\vDash F$. Later more logics will be defined thus to clarify the logic, for which the formula is valid (or true with interpretation ν), notation $\vDash_{PC} F$ (or $\nu \vDash_{PC} F$ respectively) can be used.

If there is an algorithm to check if any formula is valid, then there is also an algorithm to check if any formula is satisfiable. Actually, in order to check if formula F is not satisfiable, it must be checked if $\neg F$ is valid. In order to check if F is satisfiable, first a check if F is valid must be carried out. If F is valid, then it is satisfiable. Otherwise a check if $\neg F$ is valid

is performed. If $\neg F$ is valid, then F is not satisfiable. Otherwise it is satisfiable. This is demonstrated in the following diagram:



To test for formula validity several methods are used. The most basic one is called *truth table* method. The aim of it is to go through all the possible interpretations of the formula. If formula contains n propositional variables, each of which can be true or false, then there are exactly 2^n interpretations which must be checked. They can be listed in a table, and this is the reason for such name of the method.

Next several more complex methods are presented.

1.3.2 Hilbert-type calculus

Hilbert-type calculus is composed of many axioms and some rules. The aim of the method is to show how a formula can be constructed from the axioms, using the rules of the calculus. Here Hilbert-type calculus defined in [10] is used. There are alternative definitions, e.g. [8].

Definition 1.3.7. Hilbert-type calculus for propositional logic (denoted *HPC*) consists of axioms:

1.1. $F \supset (G \supset F)$;

1.2. $\Big(F \supset (G \supset H)\Big) \supset \Big((F \supset G) \supset (F \supset H)\Big)$;

2.1. $(F \wedge G) \supset F$;

2.2. $(F \wedge G) \supset G$;

2.3. $(F \supset G) \supset \Big((F \supset H) \supset \Big(F \supset (G \wedge H)\Big)\Big)$;

3.1. $F \supset (F \vee G)$;

3.2. $G \supset (F \vee G)$;

3.3. $(F \supset H) \supset \Big((G \supset H) \supset \Big((F \vee G) \supset H\Big)\Big)$;

4.1. $(F \supset G) \supset (\neg G \supset \neg F)$;

4.2. $F \supset \neg\neg F$;

4.3. $\neg\neg F \supset F$;

and Modus Ponens (*MP*) rule:

$$\frac{F \qquad F \supset G}{G}$$

Here F, G and H stand for any propositional formula.

Note that there are no axioms (or rules) with equivalence operator. In fact, formula $F \leftrightarrow G$ can be replaced by $(F \supset G) \wedge (G \supset F)$.

This type of calculus was formulated for the first time in [7], therefore such calculi are called Hilbert-type.

In order to check if some formula is valid, a *derivation* is constructed. A derivation of formula F in Hilbert-type calculus is a sequence of formulas $F_1, \ldots, F_n$, where $F_n \equiv F$ and for every $i \in [1, n]$, F_i is either an axiom, or obtained by applying the rules of the calculus to formulas from the set $\{F_j : j < i\}$. Thus the derivation search starts with axioms and from them new formulas are constructed by applying the rules. The process terminates successfully if formula F is finally obtained.

Here Hilbert-type derivations are presented as lists together with information on how the formula was obtained. For this purpose a substitution is used.

Definition 1.3.8. It is said that formula F is *derivable* in some Hilbert-type calculus $\mathcal{C}$ (denoted $\vdash_{\mathcal{C}} F$), if a derivation of F in $\mathcal{C}$ exists. Otherwise it is said that F is *not derivable* in $\mathcal{C}$ ($\nvdash_{\mathcal{C}} F$).

One of the core properties of the calculus are soundness and completeness. It is said that Hilbert-type calculus $\mathcal{C}$ for logic $\mathcal{L}$ is *sound* if for any formula F, if $\vdash_{\mathcal{C}} F$ then $\vDash_{\mathcal{L}} F$. It is said that Hilbert-type calculus $\mathcal{C}$ for logic $\mathcal{L}$ is *complete* if for any formula F, if $\vDash_{\mathcal{L}} F$, then $\vdash_{\mathcal{C}} F$. Only sound and complete calculi can be used to check both validity and satisfiability of any formula.

Calculus $\mathcal{C}$ is called *contradictory* if for some formula F it is true that both $\vdash_{\mathcal{C}} F$ and $\vdash_{\mathcal{C}} \neg F$. It can be proved that if a calculus is contradictory, then any formula can be derived in it. Calculus, that is sound and complete, is not contradictory. However, from the fact that calculus is not contradictory, doesn't follow that it is sound or complete.

The soundness and completeness of *HPC* is shown in [10].

Now an example of derivation in *HPC* is presented.

Example 1.3.9. A derivation of $p \supset p$ in *HPC* is as follows:

1. $\Big(p \supset \big((p \supset p) \supset p\big)\Big) \supset \Big(\big(p \supset (p \supset p)\big) \supset (p \supset p)\Big)$	Axiom *1.2*, $\{p/F, p \supset p/G, p/H\}$.
2. $p \supset \big((p \supset p) \supset p\big)$	Axiom *1.1*, $\{p/F, p \supset p/G\}$.
3. $\big(p \supset (p \supset p)\big) \supset (p \supset p)$	*MP* rule from 2 and 1.
4. $p \supset (p \supset p)$	Axiom *1.1*, $\{p/F, p/G\}$.
5. $p \supset p$	*MP* rule from 4 and 3.

Although Hilbert-type calculi are used in discussing semantics of the logic, however proof search in such calculi is not an easy task. It is hard to describe an algorithm of choosing the axioms, as can be seen in Example

1.3.9. There are several techniques suggested to tackle this problem, one of witch was first introduced in [5], and therefore is called *Gentzen-type calculus*. In Gentzen-type calculi an expression, called a *sequent* is used, thus Gentzen-type calculi are also called *sequent calculi*.

1.3.3 Sequent calculus

Definition 1.3.10. A *sequent* is an expression of the form $\Gamma \to \Delta$, where Γ and Δ are multisets of formulas and can possibly be empty. Γ is called *antecedent* and Δ is called *succedent*.

Here sequents are denoted by letter S with or without indices and capital Greek letters (Γ, Δ, Σ, Γ_1) denote multisets of formulas, which can be empty, if not mentioned otherwise. Sequents, which consist of propositional formulas only, are called propositional sequents.

The semantics of the sequent can be seen from the following definition.

Definition 1.3.11. Let's say that S is a sequent, then *corresponding formula* of S (denoted $\mathrm{Cor}(S)$) is defined as follows:

- if $S = F_1, \ldots, F_n \to G_1, \ldots, G_m$, where $n, m \geqslant 1$, then corresponding formula $\mathrm{Cor}(S) = (F_1 \wedge \ldots \wedge F_n) \supset (G_1 \vee \ldots \vee G_m)$.
- if $S = \ \to G_1, \ldots, G_m$, where $m \geqslant 1$, then $\mathrm{Cor}(S) = G_1 \vee \ldots \vee G_m$.
- if $S = F_1, \ldots, F_n \to$, where $n \geqslant 1$, then $\mathrm{Cor}(S) = \neg(F_1 \wedge \ldots \wedge F_n)$.
- if $S = \ \to$, then $\mathrm{Cor}(S) = p \wedge \neg p$ for some propositional variable p.

It is clear that if S is a propositional sequent, then $\mathrm{Cor}(S)$ is a propositional formula. This definition is similar to that given in [12].

Now it is possible to define the semantic meaning of sequent.

Definition 1.3.12. Let's say that S is a propositional sequent and ν is an interpretation of propositional formula $\mathrm{Cor}(S)$, then S is *true with interpretation* ν (denoted $\nu \vDash S$) iff $\nu \vDash \mathrm{Cor}(S)$. If $\vDash \mathrm{Cor}(S)$, then it is said that S is *valid* and denoted $\vDash S$.

Thus a sequent is true with some interpretation ν, iff when all the formulas of antecedent are true with ν, at least one formula of succedent is also true with ν.

In [5] the following calculus was defined:

Definition 1.3.13. The original Gentzen-type calculus for propositional logic (GPC_o) consists of an axiom $F \to F$, structural rules:

Weakening:

$$\frac{\Gamma \to \Delta}{F, \Gamma \to \Delta}\,(w \to) \qquad \frac{\Gamma \to \Delta}{\Gamma \to \Delta, F}\,(\to w)$$

Contraction:

$$\frac{F, F, \Gamma \to \Delta}{F, \Gamma \to \Delta}\,(c \to) \qquad \frac{\Gamma \to \Delta, F, F}{\Gamma \to \Delta, F}\,(\to c)$$

Exchange:

$$\frac{\Gamma_1, G, F, \Gamma_2 \to \Delta}{\Gamma_1, F, G, \Gamma_2 \to \Delta}\,(e \to) \qquad \frac{\Gamma \to \Delta_1, G, F, \Delta_2}{\Gamma \to \Delta_1, F, G, \Delta_2}\,(\to e)$$

logical rules:

Negation:

$$\frac{\Gamma \to \Delta, F}{\neg F, \Gamma \to \Delta}\,(\neg \to) \qquad \frac{F, \Gamma \to \Delta}{\Gamma \to \Delta, \neg F}\,(\to \neg)$$

Conjunction:

$$\frac{F, \Gamma \to \Delta}{F \wedge G, \Gamma \to \Delta}\,(\wedge \to)_1 \qquad \frac{G, \Gamma \to \Delta}{F \wedge G, \Gamma \to \Delta}\,(\wedge \to)_2$$

$$\frac{\Gamma \to \Delta, F \qquad \Gamma \to \Delta, G}{\Gamma \to \Delta, F \wedge G}\,(\to \wedge)$$

Disjunction:

$$\frac{F, \Gamma \to \Delta \qquad G, \Gamma \to \Delta}{F \vee G, \Gamma \to \Delta}\,(\vee \to)$$

$$\frac{\Gamma \to \Delta, F}{\Gamma \to \Delta, F \vee G}\,(\to \vee)_1 \qquad \frac{\Gamma \to \Delta, G}{\Gamma \to \Delta, F \vee G}\,(\to \vee)_2$$

Implication:

$$\frac{\Gamma \to \Delta, F \qquad G, \Gamma \to \Delta}{F \supset G, \Gamma \to \Delta}\,(\supset \to) \qquad \frac{F, \Gamma \to \Delta, G}{\Gamma \to \Delta, F \supset G}\,(\to \supset)$$

and the cut rule:

$$\frac{\Gamma_1 \to \Delta_1, F \qquad F, \Gamma_2 \to \Delta_2}{\Gamma_1, \Gamma_2 \to \Delta_1, \Delta_2}\,(\text{cut } F)$$

The sequent(s) above the horizontal line of the rule is (are) called the *premise(s)*. The sequent bellow the line is called the *conclusion*.

Once again, to check the validity of some sequent, derivation is constructed. Now a *derivation search tree* of the sequent S in Gentzen-type calculus is a tree of sequents, which has S at the bottom as a root and each node is either a leaf, or a conclusion of an application of some rule of the calculus in which case all the premises of the application are child nodes of

that node. S is called the initial sequent. To denote some derivation search tree, letter $\mathcal{D}$ with or without indices is used.

A *branch* of derivation search tree $\mathcal{D}$ is a subtree of $\mathcal{D}$ in which each node except the last one has exactly one child, the last node has no children and which is not a subtree of any other branch of $\mathcal{D}$ (similar definition can be found in [13]). The branch can be infinite. In that case there is no last node and each node has exactly one child. It can be noticed that every sequent of the derivation search tree belongs to at least one branch. If every branch of a derivation search tree $\mathcal{D}$ is finite, then $\mathcal{D}$ is *finite*, otherwise it is *infinite*.

If all the branches of a derivation search tree $\mathcal{D}$ of S end with axiom, it is said that $\mathcal{D}$ is a *derivation tree* (or simply a *derivation*) of S. If there exists a derivation tree of S in Gentzen-type calculus $\mathcal{C}$, it is said that S is *derivable* in $\mathcal{C}$ (denoted $\vdash_{\mathcal{C}} S$) and otherwise it is said that S is *not derivable* (denoted $\nvdash_{\mathcal{C}} S$). Formula F is derivable in sequent calculus $\mathcal{C}$ (denoted $\vdash_{\mathcal{C}} F$) iff $\vdash_{\mathcal{C}} \to F$.

Similarly to the Hilbert-type calculus, if the derivation tree of sequent S in sequent calculus is present, the reasoning about the validity of S is obvious. It starts from the axiom(s) and is continued through the applications of the rules. Due to the form of the rules, if all the premises of some application are valid, the conclusion of the application is valid too. However, the process of derivation search starts with sequent S. If S is not an axiom and it is suitable as a conclusion of some rule, then the premise(s) of the rule are inspected and the process of finding the appropriate rule is repeated to them. Thus here it is said that the rule of sequent calculus is applied to the conclusion and the premise(s) are obtained. The application of some rule is called an *inference*.

It is obvious, that usually several rules can be applied to the same sequent. To separate them, *main formula* of the inference is defined. The main formula of inferences

$$\frac{\Gamma \to \Delta, F}{\neg F, \Gamma \to \Delta}\,(\neg \to) \qquad \frac{F, \Gamma \to \Delta}{\Gamma \to \Delta, \neg F}\,(\to \neg)$$

is $\neg F$. F is called a *side formula*. Formulas from Γ and Δ are called *parametric formulas*. This definition can be extended to cover other types of inferences.

It is said that sequent calculus $\mathcal{C}$ for logic $\mathcal{L}$ is *sound* if for any sequent S, if $\vdash_{\mathcal{C}} S$ then $\vDash_{\mathcal{L}} S$. It is said that sequent calculus $\mathcal{C}$ for logic $\mathcal{L}$ is *complete* if for any sequent S, if $\vDash_{\mathcal{L}} S$, then $\vdash_{\mathcal{C}} S$. The soundness and completeness of GPC_o is proved in [5].

Now let's provide an example of derivation in GPC_o.

Example 1.3.14. A derivation in *GPC* of the formula used in Example 1.3.9 is obvious, so a derivation tree of axiom *2.3* of *HPC* is provided instead.

$$
\begin{prooftree}
\AxiomC{$F \to F$}
\RightLabel{$(\to w)$}
\UnaryInfC{$F \to F, G \wedge H$}
\RightLabel{$(\to e)$}
\UnaryInfC{$F \to G \wedge H, F$}
\RightLabel{$(w \to)$}
\UnaryInfC{$F \supset G, F \to G \wedge H, F$}
\RightLabel{$(e \to)$}
\UnaryInfC{$F, F \supset G \to G \wedge H, F$}
\AxiomC{$F \to F$}
\RightLabel{$(\to w)$}
\UnaryInfC{$F \to F, G \wedge H$}
\RightLabel{$(\to e)$}
\UnaryInfC{$F \to G \wedge H, F$}
\RightLabel{$(w \to)$}
\UnaryInfC{$H, F \to G \wedge H, F$}
\AxiomC{$G \to G$}
\RightLabel{$(w \to)$}
\UnaryInfC{$F, G \to G$}
\RightLabel{$(e \to)$}
\UnaryInfC{$G, F \to G$}
\RightLabel{$(w \to)$}
\UnaryInfC{$H, G, F \to G$}
\RightLabel{$(e \to)$}
\UnaryInfC{$G, H, F \to G$}
\AxiomC{$H \to H$}
\RightLabel{$(w \to)$}
\UnaryInfC{$F, H \to H$}
\RightLabel{$(e \to)$}
\UnaryInfC{$H, F \to H$}
\RightLabel{$(w \to)$}
\UnaryInfC{$G, H, F \to H$}
\RightLabel{$(\to \wedge)$}
\BinaryInfC{$G, H, F \to G \wedge H$}
\RightLabel{$(\supset \to)$}
\BinaryInfC{$F \supset G, H, F \to G \wedge H$}
\RightLabel{$(e \to)$}
\UnaryInfC{$H, F \supset G, F \to G \wedge H$}
\RightLabel{$(e \to)$}
\UnaryInfC{$H, F, F \supset G \to G \wedge H$}
\RightLabel{$(\supset \to)$}
\BinaryInfC{$F \supset H, F, F \supset G \to G \wedge H$}
\RightLabel{$(e \to)$}
\UnaryInfC{$F, F \supset H, F \supset G \to G \wedge H$}
\RightLabel{$(\to \supset)$}
\UnaryInfC{$F \supset H, F \supset G \to F \supset (G \wedge H)$}
\RightLabel{$(\to \supset)$}
\UnaryInfC{$F \supset G \to (F \supset H) \supset \big(F \supset (G \wedge H)\big)$}
\RightLabel{$(\to \supset)$}
\UnaryInfC{$\to (F \supset G) \supset \Big((F \supset H) \supset \big(F \supset (G \wedge H)\big)\Big)$}
\end{prooftree}
$$

However such calculus has some drawbacks too. First of all, it is not obvious how to chose the main formula in the cut rule.

Example 1.3.15. The choice of main formula in the cut rule is not obvious:

$$
\begin{prooftree}
\AxiomC{$p \to p$}
\RightLabel{$(\to w)$}
\UnaryInfC{$p \to p, q$}
\RightLabel{$(\to e)$}
\UnaryInfC{$p \to q, p$}
\AxiomC{$q \to q$}
\RightLabel{$(w \to)$}
\UnaryInfC{$p, q \to q$}
\RightLabel{$(e \to)$}
\UnaryInfC{$q, p \to q$}
\RightLabel{$(\supset \to)$}
\BinaryInfC{$p \supset q, p \to q$}
\RightLabel{$(e \to)$}
\UnaryInfC{$p, p \supset q \to q$}
\AxiomC{$q \to q$}
\RightLabel{$(\to w)$}
\UnaryInfC{$q \to q, r$}
\RightLabel{$(\to e)$}
\UnaryInfC{$q \to r, q$}
\AxiomC{$r \to r$}
\RightLabel{$(w \to)$}
\UnaryInfC{$q, r \to r$}
\RightLabel{$(e \to)$}
\UnaryInfC{$r, q \to r$}
\RightLabel{$(\supset \to)$}
\BinaryInfC{$q \supset r, q \to r$}
\RightLabel{$(e \to)$}
\UnaryInfC{$q, q \supset r \to r$}
\RightLabel{(cut q)}
\BinaryInfC{$p, p \supset q, q \supset r \to r$}
\end{prooftree}
$$

Next, there are a lot of applications of different structural rules:

Example 1.3.16. Consider the following derivation:

$$
\begin{prooftree}
\AxiomC{$p \to p$}
\RightLabel{$(w \to)$}
\UnaryInfC{$q, p \to p$}
\AxiomC{$q \to q$}
\RightLabel{$(w \to)$}
\UnaryInfC{$p, q \to q$}
\RightLabel{$(e \to)$}
\UnaryInfC{$q, p \to q$}
\RightLabel{$(\to \wedge)$}
\BinaryInfC{$q, p \to p \wedge q$}
\RightLabel{$(\to \vee)_1$}
\UnaryInfC{$q, p \to (p \wedge q) \vee (p \wedge r)$}
\AxiomC{$p \to p$}
\RightLabel{$(w \to)$}
\UnaryInfC{$r, p \to p$}
\AxiomC{$r \to r$}
\RightLabel{$(w \to)$}
\UnaryInfC{$p, r \to r$}
\RightLabel{$(e \to)$}
\UnaryInfC{$r, p \to r$}
\RightLabel{$(\to \wedge)$}
\BinaryInfC{$r, p \to p \wedge r$}
\RightLabel{$(\to \vee)_2$}
\UnaryInfC{$r, p \to (p \wedge q) \vee (p \wedge r)$}
\RightLabel{$(\vee \to)$}
\BinaryInfC{$q \vee r, p \to (p \wedge q) \vee (p \wedge r)$}
\RightLabel{$(\wedge \to)_2$}
\UnaryInfC{$p \wedge (q \vee r), p \to (p \wedge q) \vee (p \wedge r)$}
\RightLabel{$(e \to)$}
\UnaryInfC{$p, p \wedge (q \vee r) \to (p \wedge q) \vee (p \wedge r)$}
\RightLabel{$(\wedge \to)_1$}
\UnaryInfC{$p \wedge (q \vee r), p \wedge (q \vee r) \to (p \wedge q) \vee (p \wedge r)$}
\RightLabel{$(c \to)$}
\UnaryInfC{$p \wedge (q \vee r) \to (p \wedge q) \vee (p \wedge r)$}
\end{prooftree}
$$

Here application of $(c \to)$ is only needed to make sure that both p and $q \vee r$ do not disappear from the sequent, because both are needed in the derivation. Rule $(w \to)$ is applied simply to remove formulas, that are not part of the axiom. Exchange rule $(e \to)$ is applied to place the sequent in the left of the antecedent.

In fact it is possible to prove that the cut rule is not needed in the calculus. This was proved in [5] for predicate logic, which is mentioned later in Theorem 2.5.2. However, it is not hard to transform the proof to cover only propositional logic.

Theorem 1.3.17 (Gentzen Hauptsatz for propositional logic)**.** *Every derivation in GPC_o can be transformed into derivation without the cut rule.*

Later, in [9] another sequent calculus was provided, which eliminated all the structural rules.

Definition 1.3.18. Gentzen-type calculus without structural rules for propositional logic (GPC) consists of an axiom $\Gamma, F \to F, \Delta$ and the logical rules:

Negation:

$$\frac{\Gamma \to \Delta, F}{\neg F, \Gamma \to \Delta}\,(\neg \to) \qquad \frac{F, \Gamma \to \Delta}{\Gamma \to \Delta, \neg F}\,(\to \neg)$$

Conjunction:

$$\frac{F, G, \Gamma \to \Delta}{F \wedge G, \Gamma \to \Delta}\,(\wedge \to) \qquad \frac{\Gamma \to \Delta, F \quad \Gamma \to \Delta, G}{\Gamma \to \Delta, F \wedge G}\,(\to \wedge)$$

Disjunction:

$$\frac{F, \Gamma \to \Delta \quad G, \Gamma \to \Delta}{F \vee G, \Gamma \to \Delta}\,(\vee \to) \qquad \frac{\Gamma \to \Delta, F, G}{\Gamma \to \Delta, F \vee G}\,(\to \vee)$$

Implication:

$$\frac{\Gamma \to \Delta, F \quad G, \Gamma \to \Delta}{F \supset G, \Gamma \to \Delta}\,(\supset \to) \qquad \frac{F, \Gamma \to \Delta, G}{\Gamma \to \Delta, F \supset G}\,(\to \supset)$$

In this calculus the order of the formulas in antecedent and succedent is not important. This eliminates the need to apply exchange rules. The soundness and completeness of GPC is proved in [9].

Let's demonstrates how calculus GPC simplifies the derivations of sequents derived in Examples 1.3.14, 1.3.15 and 1.3.16

Example 1.3.19. A derivation in GPC of the sequent, used in Example 1.3.14.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{F, F \supset G \to G \wedge H, F \qquad \dfrac{\dfrac{F, H \to G \wedge H, F \qquad \dfrac{F, H, G \to G \quad F, H, G \to H}{F, H, G \to G \wedge H}\,(\to \wedge)}{F, H, F \supset G \to G \wedge H}\,(\supset \to)}{F, F \supset H, F \supset G \to G \wedge H}\,(\supset \to)}{F \supset H, F \supset G \to F \supset (G \wedge H)}\,(\to \supset)}{F \supset G \to (F \supset H) \supset \big(F \supset (G \wedge H)\big)}\,(\to \supset)}{\to (F \supset G) \supset \Big((F \supset H) \supset \big(F \supset (G \wedge H)\big)\Big)}\,(\to \supset)$$

Example 1.3.20. A derivation in GPC of the sequent, used in Example 1.3.15.

$$\dfrac{\dfrac{p, q, r \to r \qquad p, q \to r, q}{p, q, q \supset r \to r}(\supset \to) \qquad p, q \supset r \to r, p}{p, p \supset q, q \supset r \to r}(\supset \to)$$

Example 1.3.21. A derivation in *GPC* of the sequent, used in Example 1.3.16.

$$\dfrac{\dfrac{\dfrac{\dfrac{p, q \to p, p \wedge r \quad p, q \to q, p \wedge r}{p, q \to p \wedge q, p \wedge r}(\to \wedge) \quad \dfrac{p, r \to p \wedge q, p \quad p, r \to p \wedge q, r}{p, r \to p \wedge q, p \wedge r}(\to \wedge)}{p, q \vee r \to p \wedge q, p \wedge r}(\vee \to)}{p, q \vee r \to (p \wedge q) \vee (p \wedge r)}(\to \vee)}{p \wedge (q \vee r) \to (p \wedge q) \vee (p \wedge r)}(\wedge \to)$$

There are also other sequent calculi for propositional logic such as [2, 6].

1.3.4 Resolution method

Both of the previous methods are used to check the validity of some formula. Of course, as described in page 13, they can be used for satisfiability checking too. A method presented in this section aims to demonstrate that formula is not satisfiable. This method is called *resolution calculus*. It is clear that it can analogously be adapted for validity checking.

Let's start with several definitions.

A *literal* is an atomic formula or negation of atomic formula. In propositional logic only propositional variable is an atomic formula, so literal is a propositional variable (e.g. p) or its negation (e.g. $\neg p$). A disjunction of literals $(l_1 \vee l_2 \vee \ldots \vee l_n)$ is called a *disjunct*. A conjunction of literals $(l_1 \wedge l_2 \wedge \ldots \wedge l_n)$ is called a *conjunct*.

Definition 1.3.22. *Normal disjunctive form* (or NDF) is a disjunction of conjuncts (i.e. $F_1 \vee F_2 \vee \ldots \vee F_n$, where $F_i, i \in [1, n]$ is a conjunct).

Definition 1.3.23. *Normal conjunctive form* (or NCF) is a conjunction of disjuncts (i.e. $F_1 \wedge F_2 \wedge \ldots \wedge F_n$, where $F_i, i \in [1, n]$ is a disjunct).

It is said that two propositional formulas F and G are equivalent, if for any interpretation ν it is true that $\nu \vDash F$ iff $\nu \vDash G$. Such fact is denoted $F \equiv G$. It is not hard to prove that for any propositional formula F it is possible to find NDF (as well as NCF) G such that $F \equiv G$.

Before applying the resolution calculus a formula must be transformed into set of disjuncts. In order to do that, first a formula is transformed into NCF and then the set is composed of every disjunct of NCF. The resolution calculus contains only one rule:

$$\dfrac{F \vee p \qquad \neg p \vee G}{F \vee G}$$

Here $F \vee p$ and $\neg p \vee G$ are *premises* and $F \vee G$ is a *conclusion*. Because of commutativity of disjunction, p and $\neg p$ can be in any part of the respective premise, not necessarily at the end or at the beginning of the formula.

The application of the resolution rule produces a new disjunct, which later can be used in the derivation search. Usually the repeating literals in $F \vee G$ are omitted and only one occurrence is left. In the application of the rule F and G can be empty formulas. If both of them are empty, then it is said that *empty disjunct* (denoted $\square$) is obtained.

Definition 1.3.24. A *derivation* of disjunct F from set of disjuncts $\mathcal{S}$ in resolution calculus is a sequence $F_1, F_2, \ldots, F_n$, where $F_n = F$ and for every formula $F_i, i \in [1, n]$ are either $F_i \in \mathcal{S}$ or F_i is obtained by applying resolution rule to formulas F_k and $F_l, k, l \in [1, i)$.

It is said that disjunct F is *derivable* from set of disjuncts $\mathcal{S}$ if there is a derivation of F from $\mathcal{S}$. It is said that set of disjuncts $\mathcal{S}$ is *derivable* if $\square$ is derivable from $\mathcal{S}$.

Although a derivation is defined as a list of formulas, for convenience it will be presented as a tree.

Now to sum up, in order to derive formula F in resolution calculus, first it must be transformed into NCF. Now if the set of disjuncts of NCF is derivable, then formula F is derivable. If F is derivable in resolution calculus, then it is not satisfiable.

Example 1.3.25. An axiom *2.3* of *HPC* is valid. It is derived in sequent calculus *GPC* in Example 1.3.19. However the negation of the axiom is not satisfiable. Let's derive it in resolution calculus. First a formula must be transformed into NCF:

$$\neg\Big((F \supset G) \supset \Big((F \supset H) \supset \big(F \supset (G \wedge H)\big)\Big)\Big) =$$

$$= \neg\Big((\neg F \vee G) \supset \Big((\neg F \vee H) \supset \big(\neg F \vee (G \wedge H)\big)\Big)\Big) =$$

$$= \neg\Big(\neg(\neg F \vee G) \vee \Big(\neg(\neg F \vee H) \vee \big(\neg F \vee (G \wedge H)\big)\Big)\Big) =$$

$$= (\neg F \vee G) \wedge \neg\Big(\neg(\neg F \vee H) \vee \big(\neg F \vee (G \wedge H)\big)\Big) =$$

$$= (\neg F \vee G) \wedge \Big((\neg F \vee H) \wedge \neg\big(\neg F \vee (G \wedge H)\big)\Big) =$$

$$= (\neg F \vee G) \wedge \Big((\neg F \vee H) \wedge \big(F \wedge \neg(G \wedge H)\big)\Big) =$$

$$= (\neg F \vee G) \wedge \Big((\neg F \vee H) \wedge \big(F \wedge (\neg G \vee \neg H)\big)\Big) =$$

$$= (\neg F \vee G) \wedge (\neg F \vee H) \wedge F \wedge (\neg G \vee \neg H)$$

Thus the set of disjuncts $\mathcal{S} = \{\neg F \vee G, \neg F \vee H, F, \neg G \vee \neg H\}$. Now the derivation of $\mathcal{S}$ in resolution calculus is as follows:

$$\dfrac{\dfrac{F \qquad \neg F \vee H}{H} \qquad \dfrac{\dfrac{F \qquad \neg F \vee G}{G} \qquad \neg G \vee \neg H}{\neg H}}{\square}$$

1.3.5 Important concepts of logic

Definition 1.3.26. Suppose F and G are two propositional formulas. G is a *subformula* of F (F is a *superformula* of G), iff:

- $F = G$,
- $F = \neg F_1$ and G is a subformula of F_1, or
- $F = F_1 \wedge F_2$, $F = F_1 \vee F_2$, $F = F_1 \supset F_2$ or $F = F_1 \leftrightarrow F_2$ and G is a subformula of F_1 or F_2.

Example 1.3.27. Subformulas of formula $(p \wedge \neg q) \vee \Big((p \wedge (r \supset q)\Big)$ are: $(p \wedge \neg q) \vee \Big((p \wedge (r \supset q)\Big)$, $(p \wedge \neg q)$, p, $\neg q$, q, $p \wedge (r \supset q)$, $r \supset q$ and r.

If G is a subformula of F it can be denoted as $F(G)$. In that case formula $F(H)$ is obtained by replacing all the occurrences of subformulas G in F by H.

Example 1.3.28. Suppose $F = p \wedge (q \supset p)$, $G = q \supset p$ and $H = \neg q \vee p$. The fact that G is a subformula of F can be denoted as $F(G)$. Then $F(H) = p \wedge (\neg q \vee p)$.

Lemma 1.3.29. *These formulas are equivalent in propositional logic:*

- $\neg\neg p \equiv p$,
- $\neg(p \wedge q) \equiv \neg p \vee \neg q$,
- $\neg(p \vee q) \equiv \neg p \wedge \neg q$,
- $p \supset q \equiv \neg p \vee q$,
- $p \leftrightarrow q \equiv (p \supset q) \wedge (q \supset p)$.

Another important measure is the length of the formula. It is defined in recursive way and shows the number of logical operators in the formula.

Definition 1.3.30. The *length* of propositional formula F (denoted $\mathrm{l}(F)$) is:

- 0, if F is a propositional variable.
- $l + 1$, if F is of the form $\neg G$ and $l = \mathrm{l}(G)$.
- $l_1 + l_2 + 1$, if F is of the form $G \wedge H$, $G \vee H$, $G \supset H$ or $G \leftrightarrow H$, $l_1 = \mathrm{l}(G)$ and $l_2 = \mathrm{l}(H)$.

It is said that rule is *admissible* in some calculus if formula is derivable in the calculus using the rule iff it is derivable in the calculus without using the rule.

Most definitions presented in this subsection or in Section 1.3 can be easily extended to cover other logics, which are defined later. Thus these concepts are used later in different contexts without further notice.

1.4 Other important concepts

1.4.1 Relations

Here only binary relations (with exception of Section 2.11) are used.

Definition 1.4.1. A *binary relation* (or simply, *relation*) $\mathcal{R}$ in set $\mathcal{A}$ is any subset of $\mathcal{A} \times \mathcal{A}$.

According to the definition, a relation is some set of pairs of elements of set $\mathcal{A}$. E.g. $\{(1,2),(1,1),(3,2)\}$ is a relation in $\mathbb{N}$. However, to simplify the notation and to demonstrate that relation $\mathcal{R}$ is between two elements, instead of $(a,b) \in \mathcal{R}$ expression $a\mathcal{R}b$ is used.

The relation can have some properties, that are general for every element.

Definition 1.4.2. Suppose $\mathcal{R}$ is a binary relation in set $\mathcal{A}$, then $\mathcal{R}$ is:

- *reflexive*, iff for any $a \in \mathcal{A}$ it is true that $a\mathcal{R}a$;
- *irreflexive*, iff for any $a \in \mathcal{A}$ it is not true that $a\mathcal{R}a$;
- *complete*, iff for any $a \in \mathcal{A}$ there is b such that $a\mathcal{R}b$;
- *symmetric*, iff for any $a, b \in \mathcal{A}$ if $a\mathcal{R}b$, then $b\mathcal{R}a$;
- *asymmetric*, iff for any $a, b \in \mathcal{A}$ if $a\mathcal{R}b$, then it is not true that $b\mathcal{R}a$;
- *antisymmetric*, iff for any $a, b \in \mathcal{A}$ if $a\mathcal{R}b$ and $b\mathcal{R}a$, then $a = b$;
- *transitive*, iff for any $a, b, c \in \mathcal{A}$ if $a\mathcal{R}b$ and $b\mathcal{R}c$, then $a\mathcal{R}c$;
- *euclidean*, iff for any $a, b, c \in \mathcal{A}$ if $a\mathcal{R}b$ and $a\mathcal{R}c$, then $b\mathcal{R}c$;

Lemma 1.4.3. *It is not hard to demonstrate some properties of relations:*

- *if relation is symmetric and transitive, then it is euclidean;*
- *the relation is symmetric, transitive and complete iff it is reflexive and euclidean;*
- *if relation is reflexive, then it is complete;*
- *if relation is irreflexive and transitive, then it is asymmetric.*

If a relation is reflexive, antisymmetric and transitive, then it is a *weak partial order* relation. If a relation is irreflexive, antisymmetric and transitive, then it is a *strict partial order* relation. If relation $\mathcal{R}$ in set $\mathcal{A}$ is partial order relation and for any $a, b \in \mathcal{R}$ it is true that $a\mathcal{R}b$ or $b\mathcal{R}a$, then $\mathcal{R}$ is *total order* relation.

1.4.2 Functions and sets

Suppose $f : \mathcal{A} \to \mathcal{B}$ is a function. Point $b = f(a)$ is called an *image* of point a. An image of set $\mathcal{A}_1 \subset \mathcal{A}$ is a set $\{f(a) : a \in \mathcal{A}_1\}$.

Definition 1.4.4. Function $f : \mathcal{A} \to \mathcal{B}$ is:

- *injection*, iff the image of different points is different. I.e. for any $a_1, a_2 \in \mathcal{A}$ if $a_1 \neq a_2$, then $f(a_1) \neq f(a_2)$;
- *surjection*, iff the image of set $\mathcal{A}$ is set $\mathcal{B}$. I.e. $f(\mathcal{A}) = \mathcal{B}$;
- *bijection*, iff it is injection and surjection.

A bijection is actually a pairing of elements of two sets. It ensures that every element is paired with exactly one element of the other set and that every element has a pair. If it is possible to find a bijection between two finite sets, then both sets contain the same number of elements. Similar understanding can be extended to infinite sets. If it is possible to find a bijection between set $\mathcal{A}$ and $\mathbb{N}$, then set $\mathcal{A}$ is *countable*. There are many countable sets: $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$,.... If a set is countable, then it is possible to number all its elements in some order. However, not every set is countable. The most famous set, which is not countable is $\mathbb{R}$. If it is possible to find a bijection between set $\mathcal{A}$ and $\mathbb{R}$, then the cardinality of set $\mathcal{A}$ is called *continuum*.

Definition 1.4.5. Suppose $f : \mathcal{A} \to \mathcal{B}$ is some function. Then a function $f^{-1} : \mathcal{B} \to \mathcal{A}$ is called an *inverse function* of f if for every $x \in \mathcal{A}$ it is true that $f^{-1}(f(x)) = x$ and for every $y \in \mathcal{B}$ it is true that $f(f^{-1}(y)) = y$.

Function f is *invertible*, if there exists an inverse function of f. The following lemma can be proved:

Lemma 1.4.6. *A function is bijection iff it is invertible.*

Chapter 2

Predicate logic

2.1 The definitions

It is not always convenient to denote propositions with a single propositional variable, because the truth of the proposition could depend on the value of other parameters. Consider the proposition x *is a prime number*. It is true if $x = 3$ and false, if $x = 4$. Such propositions are called *predicates*.

Definition 2.1.1. The n-place *predicate* in set $\mathcal{A}$ is a single-valued n-argument function $f : \mathcal{A}^n \to \{\top, \bot\}$.

Example 2.1.2. Some examples of predicates:

- x and y are neighbours, if x and y are from the set of Vilnius population.
- height of x is larger than 2 meters, if x is from the set of Lithuania citizens.
- x and y are perpendicular, if x and y are from the set of lines of the plane.
- Predicate x is dividable by 5, if x is from set $\mathbb{Z}$.
- Predicate $x + y = z$, if x, y and z are from set $\mathbb{R}$.

Therefore predicate logic (denoted PR) contains three types of variables:

1. *propositional variables*, denoted by small Latin letters: p, q, r, p_1, p_2,....
2. *individual variables*, denoted by small Latin letters: x, y, z, x_1, x_2,.... Individual variables are used in predicates and sometimes are called simply *variables*.
3. *predicate variables*, denoted by capital Latin letters: $P(x)$, $Q(x, y)$, $R(x, y, z)$, $P_1(x_1, x_2)$, $P_2(z)$,.... To denote the number of arguments in predicate, an index in top right corner of the letter is used. E.g. two-place predicate $P(x, y)$ can be denoted as P^2.

The individual variables (or arguments) of the predicate denotes some element of the set in which the predicate is defined. The concrete elements of the set are called individual constants. They are usually denoted using small Latin letters a, b, c, a_1, a_2,....

Example 2.1.3. Let's consider predicate $P(x)$ defined in set $\mathbb{Z}$, which is true if x is an odd number. Then individual variable x represents any integer, and integers 2, 3, 4 are individual constants.

The interpretation of predicates can be defined in several ways. Sometimes it is convenient to list all the arguments for which the predicate is true (or false), e.g. by providing such table.

Example 2.1.4. Let's define predicate $P(x, y, z)$ in set $\{a, b, c\}$ such that $P(x, y, z) = \top$ iff $(x, y, z) \in \{(a, b, c), (a, a, a), (b, a, c), (c, a, b)\}$. Therefore, $P(c, a, b) = \top$ and $P(c, c, b) = \bot$. The table defining the predicate P is:

x	y	z
a	b	c
a	a	a
b	a	c
c	a	b

As it was mentioned earlier, predicates are propositions with variables. Of course, it is possible that predicate is true (or false) with every value of the variable. To denote that there exists such value of variable x with which predicate $P(x)$ is true, notation $\exists x P(x)$ is used. To denote that predicate $P(x)$ is true with every possible value of variable x notation $\forall x P(x)$ is used. The signs $\exists$ and $\forall$ are called *quantifiers* and can be used in front of composite formulas too. This is detailed in the definition of predicate formula.

Definition 2.1.5. *Predicate formula* is defined recursively as follows:

1. If P is n-place predicate variable and $x_1, x_2, \ldots, x_n$ are individual variables, then $P(x_1, x_2, \ldots, x_n)$ is a predicate formula. It is also called an *atomic formula*.
2. If p is a propositional variable, then p is a predicate formula. It is also called an *atomic formula*.
3. If F is a predicate formula, then $\neg F$ is also a predicate formula.
4. If F and G are predicate formulas, then $(F \wedge G)$, $(F \vee G)$, $(F \supset G)$ and $(F \leftrightarrow G)$ are predicate formulas.
5. if F is a predicate formula and x is individual variable, then $\exists x F$ and $\forall x F$ are also predicate formulas.

As well as in propositional formulas, some brackets in predicate formulas can also be omitted. Quantifiers together with $\neg$ operator have the highest priority among the operators. Atomic predicate formula is either predicate $P(x_1, x_2, \ldots, x_n)$ or propositional variable p.

Example 2.1.6. Some examples of predicate formulas:

- $\forall x \exists y \Big(\big(P(x, y) \wedge Q(y, x, z) \big) \supset \exists z R(z, x, y) \Big)$,

- $P(x, y, z) \vee \forall x \forall z\Big(Q(y, z, x) \vee \neg Q(x, y, z)\Big)$.

Definition 2.1.7. Say that predicate formula $\mathcal{Q}xG$, where $\mathcal{Q} \in \{\forall, \exists\}$ is a quantifier and G is some predicate formula, is a subformula of predicate formula F. Then the occurrence of formula G is called a *scope* of quantifier $\mathcal{Q}$ and quantifier complex $\mathcal{Q}x$.

Definition 2.1.8. An occurrence of individual variable x in formula F is called *bound* if it is in the scope of quantifier complex $\forall x$ or $\exists x$. Otherwise the considered occurrence is called *free*. If formula F has at least one free occurrence of variable x, then variable x is *free* in formula F.

Definition 2.1.9. A predicate formula is *closed*, if it does not have any free occurrence of any variable.

If for two predicate formulas $\mathcal{Q}xF$ and $\mathcal{W}xG$, where $\mathcal{Q}, \mathcal{W} \in \{\forall, \exists\}$, $\mathcal{W}xG$ is a subformula of F, then formula $\mathcal{Q}xF$ is *irregular*. E.g. formula $\forall x \exists x P(x)$ is irregular. Irregular formulas are not analysed here.

Example 2.1.10. Let's define the following propositions using predicate formulas:

1. Every user has a permission to connect to some database.
2. For every database there is a user, who can connect to it.

For formalisation let's use the following predicates:

- $U(x) = \top$ iff x is a user,
- $D(x) = \top$ iff x is a database,
- $P(x, y, z) = \top$ if x has a permission to do operation y in database z,

and individual constants:

- a — a permission to edit data,
- b — a permission to connect to the database.

The solutions are:

1. $\forall x\Big(U(x) \supset \exists y\Big(D(y) \wedge P(x, b, y)\Big)\Big)$,
2. $\forall x\Big(D(x) \supset \exists y\Big(U(y) \wedge P(y, b, x)\Big)\Big)$,

Example 2.1.11. Let's define the given propositions using predicate formulas and predicates U, R, H and E defined in set $\mathcal{M}$, where

- $\mathcal{M}$ is a set of users and reports,
- $U(x) = \top$ iff x is a user,
- $R(x) = \top$ iff x is a report,

- $H(x, y) = \top$ iff user x has report y,
- $E(x, y) = \top$ iff x is the same object as y.

The propositions are:

1. There is a user, who has no reports,
2. There are some users, who have more than one report.

The solutions are:

1. $\exists x\Big(U(x) \wedge \forall y\big(R(y) \supset \neg H(x, y)\big)\Big)$,
2. $\exists x \exists y \exists z (U(x) \wedge R(y) \wedge R(z) \wedge H(x, y) \wedge H(x, z) \wedge \neg E(y, z))$.

As well as propositional formula, predicate formula can be true or false. However the value of the formula depends not only on the value of propositional variables, but also on the definition of predicates. Moreover, formulas can contain some free variables and the value can also depend on them. Thus to interpret a predicate formula, a structure is defined.

Definition 2.1.12. Let F be a predicate formula. Let $P_1^{k_1}, P_2^{k_2}, \ldots, P_n^{k_n}$ be all the predicate variables occurring in F. Let $x_1, x_2, \ldots, x_m$ be all the free variables occurring in F. Then an *F-corresponding structure* is a multiple $\big\langle \nu, \mathcal{M}, R_1^{k_1}, R_2^{k_2}, \ldots, R_n^{k_n}, a_1, a_2, \ldots, a_m \big\rangle$, where $\nu : \mathcal{P} \to \{\top, \bot\}$ is an interpretation of propositional variables ($\mathcal{P}$ is a set of propositional variables of F), $\mathcal{M}$ is some non-empty set, $R_i^{k_i}, i \in [1, n]$ are some predicates, which are defined in $\mathcal{M}$, $a_i \in \mathcal{M}, i \in [1, m]$. Predicate $R_i^{k_i}$ is *corresponding predicate* of $P_i^{k_i}$ and individual constant a_i is a *corresponding constant* of free variable x_i.

To define which predicate of the structure corresponds to which predicate of the formula and which constant of the structure corresponds to which free variable, predicates and free variables of the formula are listed in alphabetic order. Usually predicate formulas does not contain propositional variables. If a formula F doesn't contain propositional variables, then F-corresponding structure can be presented without function ν. Similarly, if any other part of the structure is not necessary, it is omitted.

Once again the fact that formula F is true in some F-corresponding structure $\mathcal{S}$ is denoted $\mathcal{S} \vDash F$ and is defined in a recursive way:

Definition 2.1.13. Let F be some predicate formula, $P_1^{k_1}, P_2^{k_2}, \ldots, P_n^{k_n}$ — all the predicate variables occurring in F, $x_1, x_2, \ldots, x_m$ — all the free variables occurring in F and $\mathcal{S} = \big\langle \nu, \mathcal{M}, R_1^{k_1}, R_2^{k_2}, \ldots, R_n^{k_n}, a_1, a_2, \ldots, a_m \big\rangle$ — some F-corresponding structure. Then:

- if F is a propositional variable, then $\mathcal{S} \vDash F$ iff $\nu(F) = \top$.
- if $F = P_i^{k_i}(x_1, x_2, \ldots, x_{k_i})$, then let $x_{j_1}, x_{j_2}, \ldots, x_{j_m}$ be the full list of free variables in F. Let a_{l_i} be a corresponding constant of x_{j_i}, $i \in [1, m]$. Now $\mathcal{S} \vDash F$ iff $R_i^{k_i}(x_1, x_2, \ldots, x_{k_i})\{{}^{a_{l_1}}/_{x_{j_1}}, {}^{a_{l_2}}/_{x_{j_2}}, \ldots, {}^{a_{l_m}}/_{x_{j_m}}\} = \top$.

- if $F = \neg G$, then $\mathcal{S} \vDash F$ iff $\mathcal{S} \nvDash G$.
- if $F = G \wedge H$, then $\mathcal{S} \vDash F$ iff $\mathcal{S} \vDash G$ and $\mathcal{S} \vDash H$.
- if $F = G \vee H$, then $\mathcal{S} \nvDash F$ iff $\mathcal{S} \nvDash G$ and $\mathcal{S} \nvDash H$.
- if $F = G \supset H$, then $\mathcal{S} \nvDash F$ iff $\mathcal{S} \vDash G$ and $\mathcal{S} \nvDash H$.
- if $F = G \leftrightarrow H$, then $\mathcal{S} \nvDash F$ iff $\mathcal{S} \vDash G$ and $\mathcal{S} \vDash H$ or $\nu \nvDash G$ and $\nu \nvDash H$.
- if $F = \forall x G$, then $\mathcal{S} \vDash F$ iff for any $a \in \mathcal{M}$ it is true that $\mathcal{S} \vDash G\{^a/_x\}$.
- if $F = \exists x G$, then $\mathcal{S} \vDash F$ iff there is $a \in \mathcal{M}$ such that $\mathcal{S} \vDash G\{^a/_x\}$.

Definition 2.1.14. A predicate formula F is *satisfiable* if there exists an F-corresponding structure in which F is true.

Definition 2.1.15. A predicate formula F is *valid* (*tautology*), if it is true in every F-corresponding structure.

Example 2.1.16. Read the propositions and determine if they are correct (if formulas are valid). Variables x and y are integers.

1. $\forall x \exists y (x + y = 1)$,
2. $\exists x \exists y \Big((x > y) \wedge (x + y = 0) \Big)$,
3. $\exists x \forall y \Big((x + y = y) \wedge (x < 3) \Big)$,
4. $\forall x \forall y \Big((x + 3 = y) \supset (y < x) \Big)$.

The solutions are:

1. For every x there is y such that $x + y = 1$. Correct. In fact, $y = 1 - x$.
2. There is x and y such that $x > y$ and $x + y = 0$. Correct. It is enough to present one such example: let $x = 4$ and $y = -4$. It is obvious that $4 > -4$ and $4 + (-4) = 0$.
3. There is x such that for any y it is true that $x + y = y$ and $x < 3$. Correct. Once again, one such x must be found. Let $x = 0$. Then no matter what the value of y is $0 + y = y$ and $0 < 3$.
4. For any x and y if $x + 3 = y$, then $y < x$. Incorrect. To show that, one example must be found such that $x + 3 = y$ but $y \not< x$. Let $x = 1$ and $y = 4$. $1 + 3 = 4$, but $4 \not< 1$.

Example 2.1.17. Now let's analyse several formulas without any knowledge about the domain of the variables.

1. Formula $\forall x \forall y \forall z \Big(\Big(P(x, y) \wedge P(y, z) \Big) \supset P(x, z) \Big)$ is satisfiable, because it is true in structure $\langle \mathbb{R}, = \rangle$. It is obvious, that for real numbers formula $\forall x \forall y \forall z \Big(\Big((x = y) \wedge (y = z) \Big) \supset (x = z) \Big)$ is true.

2. Formula $P(x,y) \wedge \neg P(x,x) \wedge \forall x \exists y P(x,y)$ is satisfiable, because it is true in structure $\langle \mathbb{N}, <, 3, 5 \rangle$: $3 < 5$, $3 \not< 3$ and $\forall x \exists y (x < y)$.

3. Formula $\forall x P(x) \supset \exists x P(x)$ is valid. For any structure $\langle \mathcal{M}, R \rangle$ if formula $\forall x R(x)$ is true, then also formula $\exists x R(x)$ is true. Recall that according to Definition 2.1.12 $\mathcal{M} \neq \emptyset$. Therefore, if $R(x) = \top$ for every $x \in \mathcal{M}$, then for some $a \in \mathcal{M}$, $R(a) = \top$ and thus $\exists x R(x)$ is true. If, however $\forall x R(x)$ is false, then the base formula is also true in the structure.

4. Formula $\exists x \forall y \Big(P(x,x) \wedge \neg P(x,y) \Big)$ is not satisfiable. Suppose that the formula is satisfiable in structure $\langle \mathcal{M}, R \rangle$. Then there is a constant $a \in \mathcal{M}$ such that $R(a,a) = \top$. Moreover, for every $y \in \mathcal{M}$ it is not true that $R(a,y)$ and therefore $R(a,a) = \bot$. So the contradiction was obtained and therefore there is no such structure in which the base formula is true.

It is worth noting that different statements in everyday language can be denoted using the same formula. E.g. statements:

- every mushroom, which is not poisonous, is edible,
- every mushroom, which is not edible, is poisonous and
- everything, which is neither edible nor poisonous, is not mushroom at all.

are all equivalent and can be denoted using formula

$$\forall x \Big(\text{mushroom}(x) \supset \big(\neg \, \text{poisonous}(x) \supset \text{edible}(x) \big) \Big)$$

Consider syllogistic created by Aristotle, which was briefly introduced in Page 3. All the statements analysed there can be represented using predicate formulas according to this table:

(a)	Every S is P	$\forall x \big(S(x) \supset P(x) \big)$
(i)	Neither of S is P	$\forall x \big(S(x) \supset \neg P(x) \big)$
(e)	Some of S are P	$\exists x \big(S(x) \wedge P(x) \big)$
(o)	Some of S are not P	$\exists x \big(S(x) \wedge \neg P(x) \big)$

Example 2.1.18. These statements can be formalised as follows:

- Statement all the students (S) of informatics passed the algebra (A) exam is denoted as $\forall x \big(S(x) \supset A(x) \big)$.
- Statement some students (S) of informatics failed the logic (L) exam is denoted as $\exists x \big(S(x) \wedge \neg L(x) \big)$.

Thus it is possible to formally prove using predicate logic that 19 sound reasonings are really laws of logic.

The order of quantifiers in a predicate formula in some cases is essential. Formulas $\forall x \exists y P(x,y)$ and $\exists y \forall x P(x,y)$ are not equivalent. It is not hard to find a structure in which one formula is true and another one is not. E.g. structure $\langle \mathbb{N}; < \rangle$. Formula $\forall x \exists y (x < y)$ is true, but formula $\exists y \forall x (x < y)$ is false.

Lemma 2.1.19. *These formulas are equivalent in predicate logic:*

1. $\forall x \forall y F \equiv \forall y \forall x F$,
2. $\exists x \exists y F \equiv \exists y \exists x F$,
3. $\forall x F(x) \equiv \forall y F(y)$, *here* y *is not part of* $F(x)$ *and* x *is not part of* $F(y)$,
4. $\exists x F(x) \equiv \exists y F(y)$, *here* y *is not part of* $F(x)$ *and* x *is not part of* $F(y)$,
5. $\neg \forall x F \equiv \exists x \neg F$,
6. $\neg \exists x F \equiv \forall x \neg H$.

Here (and also later) $F(x)$ denotes a predicate formula with free variable x and in that case $F(y)$ denotes the same formula with every free occurrence of x replaced by y.

Definition 2.1.20. A predicate formula is in *negation normal form* if (1) it consists only of the operations $\neg$, $\wedge$ and $\vee$ and (2) the negation, if it is part of the formula, occurs only next to atomic formulas.

Example 2.1.21. Formulas $\forall x P(x) \supset \exists y \neg Q(y)$, $\neg \forall x \exists y \big(P(x) \vee \neg Q(x,y) \big)$ and $\exists x \neg \exists y \big(\neg P(x) \vee P(y) \big)$ are not in negation normal form. But formula $\forall x \exists y \forall z \Big(\big(\neg P(x,y) \wedge Q(y,z) \big) \vee \neg R(x,y) \Big)$ is.

Example 2.1.22. Let's transform formula $\neg \forall x \exists y \big(P(x,y) \supset \exists z \neg Q(x,z) \big)$ into negation normal form.

$$\neg \forall x \exists y \big(P(x,y) \supset \exists z \neg Q(x,z) \big) \equiv \neg \forall x \exists y \big(\neg P(x,y) \vee \exists z \neg Q(x,z) \big) \equiv$$

$$\equiv \exists x \neg \exists y \big(\neg P(x,y) \vee \exists z \neg Q(x,z) \big) \equiv \exists x \forall y \neg \big(\neg P(x,y) \vee \exists z \neg Q(x,z) \big) \equiv$$

$$\equiv \exists x \forall y \big(P(x,y) \wedge \neg \exists z \neg Q(x,z) \big) \equiv \exists x \forall y \big(P(x,y) \wedge \forall z Q(x,z) \big)$$

Example 2.1.23. In differential calculus a function $f(x)$ is continuous in point a if for every positive ε there is positive δ such that $|f(x) - f(a)| < \varepsilon$, if $|x - a| < \delta$.

Therefore the definition can be represented using formula

$$\forall \varepsilon \exists \delta \forall x \big((|x - a| < \delta) \supset (|f(x) - f(a)| < \varepsilon) \big)$$

Thus a function is not continuous if

$$\neg \forall \varepsilon \exists \delta \forall x \big((|x - a| < \delta) \supset (|f(x) - f(a)| < \varepsilon) \big) \equiv$$

$$\equiv \exists\varepsilon\forall\delta\exists x\neg\Big((|x-a|<\delta)\supset(|f(x)-f(a)|<\varepsilon)\Big)\equiv$$
$$\equiv \exists\varepsilon\forall\delta\exists x\neg\Big(\neg(|x-a|<\delta)\vee(|f(x)-f(a)|<\varepsilon)\Big)\equiv$$
$$\equiv \exists\varepsilon\forall\delta\exists x\Big((|x-a|<\delta)\wedge\neg(|f(x)-f(a)|<\varepsilon)\Big)\equiv$$
$$\equiv \exists\varepsilon\forall\delta\exists x\Big((|x-a|<\delta)\wedge(|f(x)-f(a)|\geqslant\varepsilon)\Big)$$

If set $\mathcal{M}$ of F-corresponding structure is finite, then it is possible to eliminate the quantifiers from the formula. Suppose predicate $P(x)$ is part of F and $\mathcal{M}=\{a_1,a_2,\ldots,a_n\}$, then quantifier $\forall$ can be eliminated from formula $\forall xP(x)$ using equivalence

$$\forall xP(x)\equiv P(a_1)\wedge P(a_2)\wedge\ldots\wedge P(a_n)$$

Similarly the quantifier $\exists$ can be eliminated from formula $\exists xP(x)$ using equivalence

$$\exists xP(x)\equiv P(a_1)\vee P(a_2)\vee\ldots\vee P(a_n)$$

Therefore, if only structures with finite set $\mathcal{M}$ (or simply *finite structures*) are analysed, then formula of predicate logic can be transformed to propositional formula.

Example 2.1.24. Let's eliminate quantifiers from formula $\exists x\forall yQ(y,y,x)$ if the set of individual constants is $\mathcal{M}=\{a,b,c\}$.

$$\exists x\forall yQ(y,y,x)=\forall yQ(y,y,a)\vee\forall yQ(y,y,b)\vee\forall yQ(y,y,c)=$$
$$=\Big(Q(a,a,a)\wedge Q(b,b,a)\wedge Q(c,c,a)\Big)\vee\Big(Q(a,a,b)\wedge Q(b,b,b)\wedge Q(c,c,b)\Big)\vee$$
$$\Big(Q(a,a,c)\wedge Q(b,b,c)\wedge Q(c,c,c)\Big)$$

However it is not enough to analyse finite structures only. This is demonstrated by the following theorem.

Theorem 2.1.25. *Formula*

$$\forall x\exists yP(x,y)\wedge\forall x\neg P(x,x)\wedge\forall x\forall y\forall z\Big(\Big(P(x,y)\wedge P(y,z)\Big)\supset P(x,z)\Big)$$

is satisfiable in infinite structure but is not satisfiable in any finite structure.

Proof. Notice, that the formula is a conjunction of three different subformulas: $\forall x\exists yP(x,y)$, $\forall x\neg P(x,x)$ and $\forall x\forall y\forall z\Big(\Big(P(x,y)\wedge P(y,z)\Big)\supset P(x,z)\Big)$. The formula is true in structure $\langle\mathbb{N},<\rangle$. In fact, for natural numbers, formulas $\forall x\exists y(x<y)$, $\forall x\neg(x<x)$ and $\forall x\forall y\forall z\Big(\Big((x<y)\wedge(y<z)\Big)\supset(x<z)\Big)$ are true.

However, in every structure with finite set the formula is false. The proof is by contradiction. Suppose that the formula is true in structure $\mathcal{S}=\langle\mathcal{M};Q(x,y)\rangle$ and $\mathcal{M}=\{a_1,a_2,\ldots,a_n\}$. Let's choose some $a_{i_1}\in\mathcal{M}$. The formula is true in the structure, therefore $\forall x\exists yQ(x,y)$ is true and there

is $a_{i_2} \in \mathcal{M}$ such that $Q(a_{i_1}, a_{i_2}) = \top$. By continuing this reasoning, it is possible to construct an infinite sequence $a_{i_1}, a_{i_2}, \ldots$ of elements of the set $\mathcal{M}$, where $Q(a_{i_j}, a_{i_{j+1}}) = \top$ for every j. Now the sequence is infinite, but the set $\mathcal{M}$ has n elements. Therefore, among the first $n+1$ elements of the sequence $(a_{i_1}, a_{i_2}, \ldots, a_{i_{n+1}})$ there are at least two, that are equal. Let's say that $a_{i_m} = a_{i_{m+k}}$.

Now, because $\forall x \forall y \forall z \Big(\big(Q(x,y) \wedge Q(y,z) \big) \supset Q(x,z) \Big)$, $Q(a_{i_m}, a_{i_{m+1}}) = \top$ and $Q(a_{i_{m+1}}, a_{i_{m+2}}) = \top$, then also $Q(a_{i_m}, a_{i_{m+2}}) = \top$. By following this reasoning, it is possible to obtain that $Q(a_{i_m}, a_{i_{m+3}}) = \top$, $Q(a_{i_m}, a_{i_{m+4}}) = \top$,... and $Q(a_{i_m}, a_{i_{m+k}}) = \top$. However, formula $\forall x \neg Q(x,x)$ is also true, and because $a_{i_m} = a_{i_{m+k}}$ a contradiction is obtained. Therefore the base formula is not satisfiable in any finite structure. □

According to the definition, a set of a structure can be of any cardinality. From Theorem 2.1.25 it is obvious that it is not enough to analyse only finite sets. However, another theorem ensures that it is enough to check only countable sets.

Theorem 2.1.26 (Löwenheim-Skolem)**.** *If formula of predicate logic is satisfiable, then it is satisfiable in some countable set.*

2.2 Normal prenex forms

As mentioned in page 20 all the formulas of propositional logic can be transformed into equivalent NDFs and NCFs. Similarly predicate formulas can be transformed into *normal prenex form.*

Definition 2.2.1. Formula is in *normal prenex form* if it is of the form $\mathcal{Q}_1 x_1 \mathcal{Q}_2 x_2 \ldots \mathcal{Q}_n x_n G$, where $\mathcal{Q}_i \in \{\forall, \exists\}, i \in [1, n]$ and G is a formula without any quantifier. G is called a matrix and $\mathcal{Q}_1 x_1 \mathcal{Q}_2 x_2 \ldots \mathcal{Q}_n x_n$ is called a prefix.

Example 2.2.2. According to the definition:

- formula $\exists x \forall y \big(P(x,x) \vee Q(y,x) \big)$ is in normal prenex form,
- formula $\exists x P(x) \supset \forall y Q(y)$ is not in normal prenex form.

If $F \equiv G$ and G is in normal prenex form, then it is said that G is a normal prenex form of F. Every formula has a normal prenex form. To transform the formula into normal prenex form the following equivalences are used:

1. $\neg \forall x F(x) \equiv \exists x \neg F(x)$,
2. $\neg \exists x F(x) \equiv \forall x \neg F(x)$,
3. $\exists x F(x) \equiv \exists y F(y)$, where y is a new variable, not part of $F(x)$,
4. $\forall x F(x) \equiv \forall y F(y)$, where y is a new variable, not part of $F(x)$,

5. $\forall x F(x) \wedge G \equiv \forall x\Big(F(x) \wedge G\Big)$, where x is not part of G,

6. $\exists x F(x) \wedge G \equiv \exists x\Big(F(x) \wedge G\Big)$, where x is not part of G,

7. $\forall x F(x) \vee G \equiv \forall x\Big(F(x) \vee G\Big)$, where x is not part of G,

8. $\exists x F(x) \vee G \equiv \exists x\Big(F(x) \vee G\Big)$, where x is not part of G,

First, all the occurrences of logical operations other than $\neg$, $\vee$ or $\wedge$ must be eliminated. This is done using equivalences mentioned in Lemma 1.3.29. Next, using equivalences 1 and 2 formula without negations in front of quantifiers must be obtained. Another step is to rename variables using equivalences 3 and 4 to ensure that every quantifier has a different variable. Finally quantifiers must be taken out using equivalences 5, 6, 7 and 8.

This procedure definitely produces a normal prenex form of any given formula of predicate logic. However to simplify the result, other equivalences can be used:

1. $\forall x F(x) \wedge \forall x G(x) \equiv \forall x\Big(F(x) \wedge G(x)\Big)$,

2. $\exists x F(x) \vee \exists x G(x) \equiv \exists x\Big(F(x) \vee G(x)\Big)$,

It must be noted that conjunction and disjunction are not interchangeable in the last two equivalences. That is, $\forall x F(x) \vee \forall x G(x) \not\equiv \forall x\Big(F(x) \vee G(x)\Big)$ and $\exists x F(x) \wedge \exists x G(x) \not\equiv \exists x\Big(F(x) \wedge G(x)\Big)$.

To prove the first one, let $F(x) = P(x)$ and $G(x) = Q(x)$. Let's analyse the structure $\langle \mathbb{N}, x < 5, x \geqslant 5\rangle$. In fact formula $\forall x(x < 5) \vee \forall x(x \geqslant 5)$ is false, but $\forall x\Big((x < 5) \vee (x \geqslant 5)\Big)$ is true.

To prove the second one, once again let $F(x) = P(x)$ and $G(x) = Q(x)$ and let's analyse the structure $\langle \mathbb{N}, x < 3, x > 10\rangle$. In fact it is true that $\exists x(x < 3) \wedge \exists x(x > 10)$, it is false that $\exists x\Big((x < 3) \wedge (x > 10)\Big)$.

Example 2.2.3. Let $F = \forall x \exists y P(x, y) \supset \forall y\Big(Q(y, y) \wedge \exists x P(y, x)\Big)$. Let's transform F into normal prenex form.

$$F = \forall x \exists y P(x, y) \supset \forall y\Big(Q(y, y) \wedge \exists x P(y, x)\Big) \equiv$$

$$\equiv \neg\forall x \exists y P(x, y) \vee \forall y\Big(Q(y, y) \wedge \exists x P(y, x)\Big) \equiv$$

$$\equiv \exists x \forall y \neg P(x, y) \vee \forall y\Big(Q(y, y) \wedge \exists x P(y, x)\Big) \equiv$$

$$\equiv \exists x \forall y \neg P(x, y) \vee \forall y \exists x\Big(Q(y, y) \wedge P(y, x)\Big) \equiv$$

$$\equiv \exists u \forall v \neg P(u, v) \vee \forall y \exists x\Big(Q(y, y) \wedge P(y, x)\Big) \equiv$$

$$\equiv \exists u \forall v \forall y \exists x\Big(\neg P(u, v) \vee \Big(Q(y, y) \wedge P(y, x)\Big)\Big)$$

Notice, that it is possible to obtain the simpler normal prenex form:

$$\exists x\forall y\neg P(x,y)\vee\forall y\exists x\Big(Q(y,y)\wedge P(y,x)\Big)\equiv$$

$$\equiv\exists x\forall v\neg P(x,v)\vee\forall y\exists x\Big(Q(y,y)\wedge P(y,x)\Big)\equiv$$

$$\equiv\forall y\Big(\exists x\forall v\neg P(x,v)\vee\exists x\Big(Q(y,y)\wedge P(y,x)\Big)\Big)\equiv$$

$$\equiv\forall y\exists x\Big(\forall v\neg P(x,v)\vee\Big(Q(y,y)\wedge P(y,x)\Big)\Big)\equiv$$

$$\equiv\forall y\exists x\forall v\Big(\neg P(x,v)\vee\Big(Q(y,y)\wedge P(y,x)\Big)\Big)$$

Example 2.2.4. Let's transform formula $\forall x\exists yP(x,y)\vee\forall x\exists yQ(x,y)$ into normal prenex form and let's try to minimise the prefix.

$$\forall x\exists yP(x,y)\vee\forall x\exists yQ(x,y)\equiv\forall x\exists yP(x,y)\vee\forall z\exists yQ(z,y)\equiv$$

$$\equiv\forall x\forall z\Big(\exists yP(x,y)\vee\exists yQ(z,y)\Big)\equiv\forall x\forall z\exists y\Big(P(x,y)\vee Q(z,y)\Big)$$

There is no algorithm to check if predicate formula is satisfiable (or valid). However for some subsets of the set of predicate formulas such algorithm exists. One of such subsets is set of formulas with one-place predicate variables only. To prove that it is always possible to check if formula with one-place predicate variables only is satisfiable, first it must be shown that any such formula can be transformed to normal prenex form with prefix $\forall\forall\ldots\forall\exists\exists\ldots\exists$. Here the complete proof is omitted and only an algorithm for such transformation is provided.

Example 2.2.5. Transform formula $\exists y\forall x\Big(\Big(P(x)\vee Q(y)\Big)\wedge\Big(R(x)\vee R(y)\Big)\Big)$ into normal prenex form with prefix $\forall\forall\ldots\forall\exists\exists\ldots\exists$.

Let's put $\forall x$ into brackets:

$$\exists y\forall x\Big(\Big(P(x)\vee Q(y)\Big)\wedge\Big(R(x)\vee R(y)\Big)\Big)\equiv$$

$$\equiv\exists y\Big(\Big(\forall xP(x)\vee Q(y)\Big)\wedge\Big(\forall xR(x)\vee R(y)\Big)\Big)$$

Let's transform the matrix into NDF:

$$\exists y\Big(\Big(\forall xP(x)\vee Q(y)\Big)\wedge\Big(\forall xR(x)\vee R(y)\Big)\Big)\equiv$$

$$\equiv\exists y\Big(\Big(\forall xP(x)\wedge\forall xR(x)\Big)\vee\Big(\forall xP(x)\wedge R(y)\Big)\vee\Big(Q(y)\wedge\forall xR(x)\Big)\vee\Big(Q(y)\wedge R(y)\Big)\Big)$$

Let's put $\exists y$ into brackets:

$$\exists y\Big(\Big(\forall xP(x)\wedge\forall xR(x)\Big)\vee\Big(\forall xP(x)\wedge R(y)\Big)\vee\Big(Q(y)\wedge\forall xR(x)\Big)\vee\Big(Q(y)\wedge R(y)\Big)\Big)\equiv$$

$$\equiv \big(\forall x P(x) \wedge \forall x R(x)\big) \vee \big(\forall x P(x) \wedge \exists y R(y)\big) \vee \big(\exists y Q(y) \wedge \forall x R(x)\big) \vee \exists y \big(Q(y) \wedge R(y)\big)$$

Let's take the quantifiers out of conjunctions. Let's start with $\forall$ and after that let's take out $\exists$:

$$\big(\forall x P(x) \wedge \forall x R(x)\big) \vee \big(\forall x P(x) \wedge \exists y R(y)\big) \vee \big(\exists y Q(y) \wedge \forall x R(x)\big) \vee \exists y \big(Q(y) \wedge R(y)\big) \equiv$$

$$\equiv \forall x \big(P(x) \wedge R(x)\big) \vee \forall x \big(P(x) \wedge \exists y R(y)\big) \vee \forall x \big(\exists y Q(y) \wedge R(x)\big) \vee \exists y \big(Q(y) \wedge R(y)\big) \equiv$$

$$\equiv \forall x \big(P(x) \wedge R(x)\big) \vee \forall x \exists y \big(P(x) \wedge R(y)\big) \vee \forall x \exists y \big(Q(y) \wedge R(x)\big) \vee \exists y \big(Q(y) \wedge R(y)\big)$$

Let's rename the variables:

$$\forall x \big(P(x) \wedge R(x)\big) \vee \forall x \exists y \big(P(x) \wedge R(y)\big) \vee \forall x \exists y \big(Q(y) \wedge R(x)\big) \vee \exists y \big(Q(y) \wedge R(y)\big) \equiv$$

$$\equiv \forall x \big(P(x) \wedge R(x)\big) \vee \forall z \exists y \big(P(z) \wedge R(y)\big) \vee \forall u \exists y \big(Q(y) \wedge R(u)\big) \vee \exists y \big(Q(y) \wedge R(y)\big)$$

Let's take the quantifiers out of the brackets:

$$\forall x \big(P(x) \wedge R(x)\big) \vee \forall z \exists y \big(P(z) \wedge R(y)\big) \vee \forall u \exists y \big(Q(y) \wedge R(u)\big) \vee \exists y \big(Q(y) \wedge R(y)\big) \equiv$$

$$\equiv \forall x \forall z \forall u \exists y \Big(\big(P(x) \wedge R(x)\big) \vee \big(P(z) \wedge R(y)\big) \vee \big(Q(y) \wedge R(u)\big) \vee \big(Q(y) \wedge R(y)\big)\Big)$$

The obtained formula is in normal prenex form.

In order to put quantifiers into brackets, the subformulas can be transformed into NDF or NCF. By doing this repeatedly, it is always possible to achieve that there are no quantifiers inside the scope of other quantifiers. This is due to the fact that formula contains one-place predicates only.

The normal prenex forms of formulas are classified into *decidable* and *undecidable* according to the form of the prefix and the type of predicate variables. Decidable means that there is an algorithm to check if a formula is satisfiable (or valid). Usually only formulas with no free variables are analysed.

A set of formulas in normal prenex form is called a *class*. Classes are defined by the pair $\langle \pi, \sigma \rangle$. Here π is any combination of elements of $\{\forall, \exists, \forall^n, \exists^n, \forall^\infty, \exists^\infty\}, n \in [2, \infty)$ and σ is a set of predicate variables. Variable σ also provides information about number of places in every predicate variable. It is obvious that π represents the type of prefix that any formula of the class has. Notation $\forall^n$ means prefix of the form $\forall x_1 \forall x_2 \ldots \forall x_n$ and notation $\exists^n$ is understood similarly. Notation $\forall^\infty$ ($\exists^\infty$) means any number (even 0) of quantifier complexes of the form $\forall x$ (respectively $\exists x$). Set σ defines the type of predicate variables, that formula may contain. It shows how many one-place, two-place, etc… predicates every formula of the class has. More formally, the class is defined as follows:

Definition 2.2.6. A pair $\langle \pi, \sigma \rangle$ denotes a class of formulas of the normal prenex form $\mathcal{Q}_1 x_1 \mathcal{Q}_2 x_2 \ldots \mathcal{Q}_n x_n G$ such that:

1. $\mathcal{Q}_i \in \{\forall, \exists\}, i \in [1, n]$,

2. G does not contain quantifiers,

3. $\pi = \mathcal{Q}_1\mathcal{Q}_2\dots\mathcal{Q}_n$,

4. there is a bijection between the set of predicate variables of G (taking into account the number of places in the predicates) and σ.

Example 2.2.7. Formula $\forall x\exists y\Big(\big(P(x)\vee P(y)\big)\wedge\big(Q(x,y)\vee R(y)\big)\Big)$ is part of the class $\Big\langle\forall\exists,\{P_1^1,P_2^1,P_3^2\}\Big\rangle$, because it has two one-placed and one two-placed predicate variables and the prefix $\forall\exists$.

Suppose π_1 and π_2 are the two representations of the prefix in definition of the class. It is said that π_1 is *part* of π_2, if it is possible to obtain π_1 from π_2 using finite number of operations:

1. removing symbol $\forall$ or $\exists$,

2. replacing symbol $\forall^\infty$ by $\forall^n$ or symbol $\exists^\infty$ by $\exists^n$,

3. replacing symbol $\forall^n$ by $\forall^m$ or symbol $\exists^n$ by $\exists^m$, if $m < n$.

Definition 2.2.8. It is said that $\langle\pi_1,\sigma_1\rangle \leqslant \langle\pi_2,\sigma_2\rangle$ (the later class is *larger* than the former one) if:

1. $\pi_1 = \pi_2$ or π_1 is part of π_2,

2. there is a bijection between σ_1 and subset of σ_2.

It is easy to see that if some class is undecidable, then any larger class is undecidable too. Now let's list several minimal undecidable classes:

- $\Big\langle\exists\forall\exists,\{P^2,Q_1^1,Q_2^1,\dots\}\Big\rangle$;
- $\Big\langle\exists^3\forall,\{P^2,Q_1^1,Q_2^1,\dots\}\Big\rangle$;
- $\Big\langle\forall^\infty\exists^3\forall,\{P^2\}\Big\rangle$;
- $\Big\langle\exists^\infty\forall,\{P^2\}\Big\rangle$;
- $\Big\langle\exists\forall\exists^\infty,\{P^2\}\Big\rangle$;
- $\Big\langle\exists^3\forall^\infty,\{P^2\}\Big\rangle$;
- $\Big\langle\forall^\infty\exists\forall\exists,\{P^2\}\Big\rangle$;
- $\Big\langle\exists\forall^\infty\exists,\{P^2\}\Big\rangle$;
- $\Big\langle\exists\forall\exists\forall^\infty,\{P^2\}\Big\rangle$.

It should be noted that from the fact that class $\Big\langle\exists\forall\exists,\{P^2,Q_1^1,Q_2^1,\dots\}\Big\rangle$ is undecidable follows that any larger class is also undecidable. Thus the

statement that such class is undecidable means that any formula with following properties is undecidable:

- prefix of the formula is $\exists\forall\exists$ or $\exists\forall\exists$ is part of the prefix of the formula and
- formula contains at least one two-place predicate and any number of one-place predicates.

A class $\langle\pi,\sigma\rangle$ is decidable if:

- σ composes of one-place predicate variables only;
- $\pi = \forall^{\infty}\exists^{\infty}$;
- $\pi = \forall^{\infty}\exists^{2}\forall^{\infty}$;
- $\langle\pi,\sigma\rangle \leqslant \langle\pi',\sigma'\rangle$, where $\langle\pi',\sigma'\rangle$ is decidable.

2.3 Predicate logic with function symbols

Any expression that can occur as the parameter of the predicate, is called, a *term*. Individual constants and individual variables are terms. However, let's extend the notion of term by including functions. Suppose predicates are defined in set $\mathcal{M}$. Then only functions of the form $f : \mathcal{M}^n \to \mathcal{M}$ are analysed for any $n \in \mathbb{N}\backslash\{0\}$. Functions are denoted by small Latin letters f, g, h, f_1,... and as well as for predicate variables, the number of arguments of the function can be presented as the top-right index: f^n. Terms are denoted using small t with or without index.

Suppose, a set of individual constants consists of days of the year. There are some examples of functions defined in this set: $y = \text{tomorrow}(x)$ (if $x = 14^{th}$ *of March*, then $\text{tomorrow}(x) = 15^{th}$ *of March*), $y = \text{yesterday}(x)$, $y = \text{one\_week\_later}(x)$, $y = \text{one\_month\_later}(x)$.

Definition 2.3.1. A *term* is defined as follows:

- Individual constant is a term.
- Individual variable is a term.
- If f^n is function symbol and $t_1, t_2, \ldots, t_n$ are terms, then $f(t_1, t_2, \ldots, t_n)$ is also a term.

Example 2.3.2. Let $\mathcal{M} = \{a, b, c\}$, $f(x)$ is one-place function symbol. Then the following expressions are terms: a, b, c, x, y, z, $f(x)$, $f(a)$, $f(c)$, $f\big(f(x)\big)$, $f\big(f(a)\big)$, $f\Big(f\big(f(y)\big)\Big)$,....

Now let's define predicate formula with function symbols.

Definition 2.3.3. *Formula* is defined recursively as follows:

1. If P is n-place predicate variable and $t_1, t_2, \ldots, t_n$ are terms, then $P(t_1, t_2, \ldots, t_n)$ is a formula.

2. If p is a propositional variable, then p is a formula.
3. If F is a formula, then $\neg F$ is also a formula.
4. If F and G are formulas, then $(F \wedge G)$, $(F \vee G)$, $(F \supset G)$ and $(F \leftrightarrow G)$ are formulas.
5. if F is a formula and x is individual variable, then $\exists x F$ and $\forall x F$ are also formulas.

Suppose, a set of individual constants consists of all the lines in the plain. Function $f(x)$ returns a line y which is perpendicular to line x and crosses the centre of coordinate system. Say, predicate $L(x, y) = \top$ iff lines x and y are parallel. Then formula $L(x, f(y)) = \top$, iff line x is parallel to a line, which is perpendicular to y and crosses the centre of coordinate system.

The definition of expression $F(x)$ can be extended to cover terms. Suppose x is a free variable in formula $F(x)$ and t is a term. Formula $F(t)$ is obtained from $F(x)$ by replacing all the free occurrences of x by term t.

Example 2.3.4. If $F(x) = \forall x\Big(P(x) \vee Q(x)\Big) \wedge R(x, x)$ and $t = f\Big(a, g(y)\Big)$, then $F(t) = \forall x\Bigg(P\Big(x\Big) \vee Q\Big(x\Big)\Bigg) \wedge R\Big(f\Big(a, g(y)\Big), f\Big(a, g(y)\Big)\Big)$.

It is possible to do without function symbols in predicate logic. This is just a more convenient (natural) way to formalise expressions, formulas with function symbols are shorter. E.g. instead of $y = f(x)$ a predicate $P(x, y)$ could be defined such that $P(x, y) = \top$ iff $y = f(x)$.

Of course, to interpret predicate formulas with function symbols a different structure is needed. Such structure must cover interpretation of the functions.

Definition 2.3.5. Let F be a predicate formula with function symbols. Let $P_1^{k_1}, P_2^{k_2}, \ldots, P_n^{k_n}$ be all the predicate variables occurring in F. Let $x_1, x_2, \ldots, x_m$ be all the free variables occurring in F. Let $f_1^{l_1}, f_2^{l_2}, \ldots, f_u^{l_u}$ be all the function symbols occurring in F. Then multiple

$$\Big\langle \nu, \mathcal{M}, R_1^{k_1}, R_2^{k_2}, \ldots, R_n^{k_n}, a_1, a_2, \ldots, a_m, g_1^{l_1}, g_2^{l_2}, \ldots, g_u^{l_u} \Big\rangle$$

is called an *F-corresponding structure* if $\nu : \mathcal{P} \to \{\top, \bot\}$ is an interpretation of propositional variables ($\mathcal{P}$ is a set of propositional variables of F), $\mathcal{M}$ is some non-empty set, $R_i^{k_i}, i \in [1, n]$ are some predicates, which are defined in $\mathcal{M}$, $a_i \in \mathcal{M}, i \in [1, m]$, $g_i^{l_i} : \mathcal{M}^{l_i} \to \mathcal{M}, i \in [1, u]$ are some functions. Predicate $R_i^{k_i}$ is *corresponding predicate* of $P_i^{k_i}$, individual constant a_i is a *corresponding constant* of free variable x_i and function $g_i^{l_i}$ is *corresponding function* of $f_i^{l_i}$.

Once again, a concept of $\mathcal{S} \vDash F$ must be redefined, however it should be obvious how to change Definition 2.1.13 to incorporate function symbols. The definitions of satisfiable and valid formulas are analogous to Definitions 2.1.14 and 2.1.15.

Now to better understand the definitions let's analyse several examples.

Example 2.3.6. Formula $\forall x \exists y \Big(P\big(x, y, f(x)\big) \wedge Q\big(y, y, y\big) \Big)$ is satisfiable. It is true in structure $\langle \mathbb{N}, x + y = z, xy = z, x + 1 \rangle$. That is, a formula $\forall x \exists y \big((x + y = x + 1) \wedge (yy = y) \big)$ is true. For any x if $y = 1$, then both parts of the conjunction are true.

Example 2.3.7. Formula

$$\forall x \neg P\big(x, x\big) \wedge \forall x P\big(y, f(x)\big) \wedge \forall x \exists y \Big(P\big(f(x), y\big) \wedge P\big(y, f(f(x))\big) \Big)$$

is satisfiable, because it is true in structure $\langle \mathbb{N}, x < y, 1, x + 2 \rangle$. That is, $\forall x \neg (x < x) \wedge \forall x (1 < x + 2) \wedge \forall x \exists y \big((x + 2 < y) \wedge (y < x + 4) \big)$ is true for natural numbers.

Example 2.3.8. Is formula $\forall x \Big(P\big(f(x), x\big) \wedge P\big(x, g(x)\big) \Big) \supset \forall x P\big(f(x), g(x)\big)$ valid?

To answer this question positively, it must be proved that formula is true in every structure. To answer this question negatively, it is enough to find one structure, in which this formula is false. Actually, this formula is false in structure $\langle \mathcal{M}, P', f', g' \rangle$, where:

- $\mathcal{M} = \{1, 2, 3\}$,
- $P'(x, y) = \top$ iff $(x, y) \in \{(1, 1), (1, 2), (2, 3), (3, 3)\}$,
- $f'(1) = 1,\ f'(2) = 1,\ f'(3) = 3$,
- $g'(1) = 2,\ g'(2) = 3,\ g'(3) = 3$.

Formula $\forall x \Big(P'\big(f'(x), x\big) \wedge P'\big(x, g'(x)\big) \Big)$ is true, but $\forall x P'\big(f'(x), g'(x)\big)$ is false, because $P'\big(f'(2), g'(2)\big) = P'(1, 3) = \bot$.

Example 2.3.9. Is formula $\forall x P\big(x, f(x)\big) \supset P\big(f(a), f(f(a))\big)$ valid?

Actually it is, and in order to prove that let's assume contrary, that there is a structure $\langle \mathcal{M}, P', f' \rangle$ such that $\forall x P'\big(x, f'(x)\big) \supset P'\big(f'(a), f'(f'(a))\big)$ is false. That is, $\forall x P'\big(x, f'(x)\big)$ is true and $P'\big(f'(a), f'(f'(a))\big)$ is false. The former formula, is true for every x, therefore x can be changed to $f'(a)$. After the change, formula $P'\big(f'(a), f'(f'(a))\big)$ is obtained, which is true according to the former formula, but false according to the latter one. This is the contradiction, therefore the base formula is valid.

By using function symbols it is possible to remove $\exists$ quantifier. The process of finding equivalent formula without $\exists$ quantifier is called *skolemization* and was described in 1920 by Norwegian logician Th. Skolem (1887–1963).

Before skolemization, a formula must be transformed into normal prenex form. Suppose $F = \mathcal{Q}_1 x_1 \mathcal{Q}_2 x_2 \ldots \mathcal{Q}_n x_n G(x_1, x_2, \ldots, x_n)$ is such formula (i.e. $\mathcal{Q}_i \in \{\forall, \exists\}, i \in [1, n]$, formula $G(x_1, x_2, \ldots, x_n)$ has no quantifiers and $x_1, x_2, \ldots, x_n$ is the full list of free variables in G). Skolemization can be

described as follows. Suppose $\mathcal{Q}_r = \exists$ is the first such quantifier in F. That is, $\mathcal{Q}_1 = \mathcal{Q}_2 = \cdots = \mathcal{Q}_{r-1} = \forall$. Let

$$F' = \pi G\Big(x_1, x_2, \ldots, x_{r-1}, f(x_1, x_2, \ldots, x_{r-1}), x_{r+1}, x_{r+2}, \ldots, x_n\Big)$$

where prefix $\pi = \forall x_1 \forall x_2 \ldots \forall x_{r-1} \mathcal{Q}_{r+1} x_{r+1} \mathcal{Q}_{r+2} x_{r+2} \ldots \mathcal{Q}_n x_n$ and f is a new function variable, which is not part of G. If $r = 1$, i.e. $\exists$ is the first quantifier of F, then let $F' = \mathcal{Q}_2 x_2 \mathcal{Q}_3 x_3 \ldots \mathcal{Q}_n x_n G(a, x_2, x_3 \ldots, x_n)$, where a is some new individual constant, which is not part of G. Such process is continued until there is no $\exists$ quantifier in F'.

Example 2.3.10. Let's skolemize formula

$$\exists x_1 \exists x_2 \forall y_1 \forall y_2 \exists x_3 \forall y_3 \exists x_4 G(x_1, x_2, y_1, y_2, x_3, y_3, x_4)$$

Let a and b be new constants, that are not part of G. Let $f(y_1, y_2)$ and $g(y_1, y_2, y_3)$ be new function symbols. Then the skolemized formula is:

$$\forall y_1 \forall y_2 \forall y_3 G\Big(a, b, y_1, y_2, f(y_1, y_2), y_3, g(y_1, y_2, y_3)\Big)$$

Suppose, F is some closed formula. Let's do the following:

1. transform it into normal prenex form,
2. skolemize it,
3. remove all the quantifiers $\forall$,
4. transform it into NCF.

The resulting formula is called a *standard form* of formula F. Although there are no quantifiers in the standard form, it is assumed that all the free variables are bounded by quantifier $\forall$.

Example 2.3.11. Let's transform formula

$$\forall x\Big(\big(P(x) \wedge Q(x)\big) \supset \exists y\big(R(x, y) \wedge S(y)\big)\Big)$$

into standard form.

$$\forall x\Big(\big(P(x) \wedge Q(x)\big) \supset \exists y\big(R(x, y) \wedge S(y)\big)\Big) \equiv$$

$$\equiv \forall x \exists y\Big(\big(P(x) \wedge Q(x)\big) \supset \big(R(x, y) \wedge S(y)\big)\Big) \equiv$$

$$\equiv \forall x\Big(\Big(P\big(x\big) \wedge Q\big(x\big)\Big) \supset \Big(R\big(x, f(x)\big) \wedge S\big(f(x)\big)\Big)\Big) \equiv$$

$$\equiv \Big(P\big(x\big) \wedge Q\big(x\big)\Big) \supset \Big(R\big(x, f(x)\big) \wedge S\big(f(x)\big)\Big) \equiv$$

$$\equiv \neg\Big(P\big(x\big) \wedge Q\big(x\big)\Big) \vee \Big(R\big(x, f(x)\big) \wedge S\big(f(x)\big)\Big) \equiv$$

$$\equiv \Big(\neg P\big(x\big) \vee \neg Q\big(x\big)\Big) \vee \Big(R\big(x, f(x)\big) \wedge S\big(f(x)\big)\Big) \equiv$$

$$\equiv \Big(\neg P\big(x\big) \vee \neg Q\big(x\big) \vee R\big(x, f(x)\big)\Big) \wedge \Big(\neg P\big(x\big) \vee \neg Q\big(x\big) \vee S\big(f(x)\big)\Big)$$

Formulas with function symbols can also be classified into decidable and undecidable classes. Recall that formulas must be in normal prenex form.

For formulas with function symbols different notation is used. A class is presented as $\pi(\text{pred:}i_1, i_2, \ldots, i_n; \text{funkc:}j_1, j_2, \ldots, j_m)$. Here π is a prefix of a formula, $i_k, k \in [1, n]$ means that formula must contain i_k k-place predicate variables. The values of $j_l, l \in [1, m]$ means that formula must contain j_l l-place function variables. If $i_k = \infty$, it means that formula may contain any number of k-place predicate variables. Notation j_l is interpreted analogously. If instead of all the i_k (j_l) a word *any* is written, then any number of any type of predicate (respectively function) variables may be present in a formula. If instead of concrete prefix, letter π is written, formula may contain any prefix.

Example 2.3.12. $\pi(\text{pred:}\infty; \text{funkc:}1)$ denotes a class of normal prenex form formulas with any prefix, any number of one-place predicate variables and a single one-place function symbol.

American scientist Y. Gurevichius proved that class $\pi(\text{pred:}\infty; \text{funkc:}\infty)$ is maximal decidable and $\exists\exists(\text{pred:}0, 1; \text{funkc:}1)$ and $\exists\exists(\text{pred:}1; \text{funkc:}0, 1)$ are minimal undecidable classes.

In some cases equality predicate is incorporated in syntax of predicate logic. Therefore formulas with equality predicate are also classified according to decidability:

Maximal decidable classes:

- $\pi(=; \text{pred:}\infty; \text{funkc:}1)$,
- $\forall^*(=; \text{pred:any}; \text{funkc:any})$,
- $\forall^*\exists\forall^*(=; \text{pred:any}; \text{funkc:}1)$,
- $\forall^*\exists^*(=; \text{pred:any}; \text{constants:}\infty)$.

Minimal undecidable classes:

- $\exists\exists\forall(=; \text{pred:}\infty, 1)$,
- $\exists\exists\forall(=; \text{pred:}0, 1; \text{constants:}\infty)$,
- $\exists\exists\forall^*(=; \text{pred:}0, 1)$,
- $\forall^*\exists\exists\forall(=; \text{pred:}0, 1)$,
- $\exists(=; \text{funkc:}2)$,
- $\exists(=; \text{funkc:}0, 1)$.

2.4 Hilbert-type calculus

Definition 2.4.1. It is said that term t is *free* with respect to variable x in formula $F(x)$, if for any individual variable y in t, x doesn't occur in the scope of $\forall y$, nor of $\exists y$ in formula $F(x)$.

Example 2.4.2. Let $F = \forall y \exists z \forall u \big(P(y,z) \vee Q(x,v)\big)$. Terms $f\big(b, f(v,w)\big)$, $g\big(f(x),w\big)$, x, v are free with respect to x in F. Terms $f(x,y)$, $g\big(u, f(a,b)\big)$, z are not free with respect to x in F.

Hilbert-type predicate calculus is similar to propositional Hilbert-type calculus *HPC*.

Definition 2.4.3. Hilbert-type calculus for predicate logic (denoted *HPR*) consists of axioms of *HPC* and:

5.1 $\forall x F(x) \supset F(t)$,

5.2 $F(t) \supset \exists x F(x)$.

Here, t is a term, which is free with respect to variable x in formula $F(x)$.

The rules of *HPR* are:

$$\frac{F \qquad F \supset G}{G}\, MP \qquad \frac{G \supset F(y)}{G \supset \forall x F(x)}\, \forall \qquad \frac{F(y) \supset G}{\exists x F(x) \supset G}\, \exists$$

Here y is a free variable in $F(y)$, but it does not occur in $G \supset \forall x F(x)$ freely. Moreover, y must be free with respect to x in formula $F(x)$. Notice, that y can possibly be equal to x.

The additional requirement for term t in axioms 5.1 and 5.2 is necessary. Without it, it would be possible to derive false statements from the correct ones. E.g. $\forall x \exists y (x \neq y)$ is a satisfiable formula (e.g. in set $\mathbb{N}$), however $\exists y (y \neq y)$ is not and therefore, formula $\forall x \exists y (x \neq y) \supset \exists y (y \neq y)$ is not valid and should not be an axiom of this calculus. Actually, in this case y is not free with respect to x in formula $\forall x \exists y (x \neq y)$.

The additional requirements for variable y in rules $\forall$ and $\exists$ are also necessary. This could also be demonstrated with an example, which does not respect the requirement:

$$\frac{(x > 7) \supset (x > 3)}{(x > 7) \supset \forall x (x > 3)}$$

Although the premise is valid, the conclusion is not. This is because x occurs freely in the conclusion.

In a separate case, when G is not present, rule $\forall$ can be transformed into:

$$\frac{F(y)}{\forall x F(x)}\, \forall'$$

Let's analyse an example of derivation in *HPR*.

Example 2.4.4. Let's show, that from formula $\forall x \forall y P(x,y)$ another formula $\forall y \forall x P(x,y)$ is derivable. Let's construct a derivation:

1.	$\forall x \forall y P(x,y)$	An assumption.
2.	$\forall x \forall y P(x,y) \supset \forall y P(a,y)$	Axiom *5.1.*
3.	$\forall y P(a,y)$	*MP* rule from 1 and 2.
4.	$\forall y P(a,y) \supset P(a,b)$	Axiom *5.1.*
5.	$P(a,b)$	*MP* rule from 3 and 4.
6.	$\forall x P(x,b)$	$\forall$ rule from 5.
7.	$\forall y \forall x P(x,y)$	$\forall$ rule from 6.

It is known that *HPR* is sound and complete. However, let's demonstrate a weaker statement.

Theorem 2.4.5. *Hilbert-type predicate calculus HPR is not contradictory.*

Proof. Let's define a transformation operator $\mathrm{Tr}(F)$, which transforms predicate formula F into propositional formula:

- $\mathrm{Tr}\Big(P(t_1, t_2, \ldots, t_n)\Big) = p$, that is to an atomic formula a new propositional variable is assigned,
- $\mathrm{Tr}(\neg F) = \neg\,\mathrm{Tr}(F)$,
- $\mathrm{Tr}(F \wedge G) = \mathrm{Tr}(F) \wedge \mathrm{Tr}(G)$,
- $\mathrm{Tr}(F \vee G) = \mathrm{Tr}(F) \vee \mathrm{Tr}(G)$,
- $\mathrm{Tr}(F \supset G) = \mathrm{Tr}(F) \supset \mathrm{Tr}(G)$,
- $\mathrm{Tr}\Big(\forall x F(x)\Big) = \mathrm{Tr}\Big(F(x)\Big)$,
- $\mathrm{Tr}\Big(\exists x F(x)\Big) = \mathrm{Tr}\Big(F(x)\Big)$.

This operator has the following properties:

- If predicate formula F is an axiom of *HPR*, then $\mathrm{Tr}(F)$ is a valid formula.
- If $\mathrm{Tr}(F)$ is a valid formula and G is obtained from F using rules $\forall$, $\forall'$ or $\exists$, then $\mathrm{Tr}(G)$ is also a valid formula, because $\mathrm{Tr}(F) = \mathrm{Tr}(G)$.
- If $\mathrm{Tr}(F)$ and $\mathrm{Tr}(F \supset G)$ are both valid, then $\mathrm{Tr}(G)$ is also valid.

Therefore, if some formula F is derivable in Hilbert-type predicate calculus, then $\mathrm{Tr}(F)$ is a valid formula. Therefore, it is impossible to derive formula $\neg F$, because $\mathrm{Tr}(\neg F) = \neg\,\mathrm{Tr}(F)$ is not satisfiable formula. □

2.5 Sequent calculus

A definition of sequent is the same as in propositional logic, except that the formulas now are from predicate logic.

Definition 2.5.1. A sequent calculus for predicate logic (denoted GPR_o) is obtained by adding rules for quantifiers into calculus GPC_o:

$$\frac{F(z), \Gamma \to \Delta}{\exists x F(x), \Gamma \to \Delta}\,(\exists \to) \qquad \frac{\Gamma \to \Delta, \exists x F(x), F(t)}{\Gamma \to \Delta, \exists x F(x)}\,(\to \exists)$$

$$\frac{F(t), \forall x F(x), \Gamma \to \Delta}{\forall x F(x), \Gamma \to \Delta}\,(\forall \to) \qquad \frac{\Gamma \to \Delta, F(z)}{\Gamma \to \Delta, \forall x F(x)}\,(\to \forall)$$

Here z is a new variable, which doesn't belong to Γ, Δ, $\exists x F(x)$ nor $\forall x F(x)$ and t is a term, which is free with respect to x in formula $F(x)$.

It have already been mentioned that the main result of Gentzen was the proof of cut-elimination theorem for predicate logic.

Theorem 2.5.2 (Gentzen Hauptsatz)**.** *If all the free and bound variables in some sequent are denoted using different letters, then the sequent is derivable in predicate sequent calculus GPR_o iff it is derivable without using the cut rule.*

However, later more convenient calculi without structural rules were developed. To obtain such calculus let's add the same quantifier rules into calculus for propositional logic *GPC*.

Definition 2.5.3. Gentzen-type calculus without structural rules for predicate logic (*GPR*) is obtained by adding rules for quantifiers into calculus *GPC*:

$$\frac{F(z), \Gamma \to \Delta}{\exists x F(x), \Gamma \to \Delta}\,(\exists \to) \qquad \frac{\Gamma \to \Delta, \exists x F(x), F(t)}{\Gamma \to \Delta, \exists x F(x)}\,(\to \exists)$$

$$\frac{F(t), \forall x F(x), \Gamma \to \Delta}{\forall x F(x), \Gamma \to \Delta}\,(\forall \to) \qquad \frac{\Gamma \to \Delta, F(z)}{\Gamma \to \Delta, \forall x F(x)}\,(\to \forall)$$

Here z is a new variable, which doesn't belong to Γ, Δ, $\exists x F(x)$ nor $\forall x F(x)$ and t is a term, which is free with respect to x in formula $F(x)$.

Order of formulas in multisets is not important.

Example 2.5.4. Let's show that sequent $\to \forall x \forall y P(x,y) \supset \forall y \forall x P(x,y)$ is derivable in calculus *GPR*. Recall, that this sequent is similar to the case analysed in Example 2.4.4.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{P(x_1,y_1), \forall y P(x_1,y), \forall x \forall y P(x,y) \to P(x_1,y_1)}{\forall y P(x_1,y), \forall x \forall y P(x,y) \to P(x_1,y_1)}\,(\forall \to)}{\forall x \forall y P(x,y) \to P(x_1,y_1)}\,(\forall \to)}{\forall x \forall y P(x,y) \to \forall x P(x,y_1)}\,(\to \forall)}{\forall x \forall y P(x,y) \to \forall y \forall x P(x,y)}\,(\to \forall)}{\to \forall x \forall y P(x,y) \supset \forall y \forall x P(x,y)}\,(\to \supset)$$

Example 2.5.5. Let's derive sequent: $\to \exists x \forall y P(x,y) \supset \forall y \exists x P(x,y)$ in calculus *GPR*.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{P(x_1,y_1), \forall y P(x_1,y) \to \exists x P(x,y_1), P(x_1,y_1)}{P(x_1,y_1), \forall y P(x_1,y) \to \exists x P(x,y_1)}\,(\to \exists)}{\forall y P(x_1,y) \to \exists x P(x,y_1)}\,(\forall \to)}{\forall y P(x_1,y) \to \forall y \exists x P(x,y)}\,(\to \forall)}{\exists x \forall y P(x,y) \to \forall y \exists x P(x,y)}\,(\exists \to)}{\to \exists x \forall y P(x,y) \supset \forall y \exists x P(x,y)}\,(\to \supset)$$

Example 2.5.6. Now a derivation of a well known tautology of predicate logic is presented: $\to \big(\neg\forall x P(x) \supset \exists x \neg P(x)\big) \wedge \big(\exists x \neg P(x) \supset \neg\forall x P(x)\big)$.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{P(z) \to \exists x \neg P(x), P(z)}{\to \exists x \neg P(x), \neg P(z), P(z)}(\to \neg)}{\to \exists x \neg P(x), P(z)}(\to \exists)}{\to \exists x \neg P(x), \forall x P(x)}(\to \forall)}{\neg\forall x P(x) \to \exists x \neg P(x)}(\neg \to)}{\to \neg\forall x P(x) \supset \exists x \neg P(x)}(\to \supset) \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{P(z), \forall x P(x) \to P(z)}{\forall x P(x) \to P(z)}(\forall \to)}{\forall x P(x), \neg P(z) \to}(\neg \to)}{\forall x P(x), \exists x \neg P(x) \to}(\exists \to)}{\exists x \neg P(x) \to \neg\forall x P(x)}(\to \neg)}{\to \exists x \neg P(x) \supset \neg\forall x P(x)}(\to \supset)$$

$$\dfrac{\to \neg\forall x P(x) \supset \exists x \neg P(x) \qquad \to \exists x \neg P(x) \supset \neg\forall x P(x)}{\to \big(\neg\forall x P(x) \supset \exists x \neg P(x)\big) \wedge \big(\exists x \neg P(x) \supset \neg\forall x P(x)\big)}(\to \wedge)$$

One of the most popular predicates in applications of logic is equality. Instead of traditional predicate notation, equality predicate is usually denoted as $t_1 = t_2$, where t_1 and t_2 are terms.

Definition 2.5.7. Sequent calculus with equality predicate (denoted as *GPR*=) is obtained from *GPR* by adding axiom $\Gamma \to \Delta, t = t$ and two additional rules:

$$\dfrac{t_1 = t_2, \Gamma\{t_2/t_1\} \to \Delta\{t_2/t_1\}}{t_1 = t_2, \Gamma \to \Delta}(=_1) \qquad \dfrac{t_1 = t_2, \Gamma\{t_1/t_2\} \to \Delta\{t_1/t_2\}}{t_1 = t_2, \Gamma \to \Delta}(=_2)$$

Here t, t_1 and t_2 are terms.

An occurrence of term in a formula is called the *main* one, if it does not contain any bound occurrence of variable.

Definition 2.5.8. A sequent calculus is called *minus-normal*, if term t in applications of rules $(\to \exists)$ and $(\forall \to)$ is a main term and part of some formula of the conclusion of the inference. If the conclusion does not have any main terms, then t is a new constant.

The equivalence of calculi *GPR* and minus-normal calculus was proved in 1963 by S. Kanger.

It is already mentioned in page 38 that class of predicate formulas without function symbols $\langle \forall^\infty \exists^\infty, \sigma \rangle$, where σ is any set of predicates, is decidable. Let's prove that using minus-normal predicate calculus.

Theorem 2.5.9. *Class of formulas without function symbols with prefix* $\forall^\infty \exists^\infty$ *is decidable.*

Proof. Suppose F is some formula without function symbols,

$$F = \forall x_1 \forall x_2 \ldots \forall x_n \exists y_1 \exists y_2 \ldots \exists y_m G(x_1, x_2, \ldots, x_n, y_1, y_2, \ldots, y_m)$$

Here $G(x_1, x_2, \ldots, x_n, y_1, y_2, \ldots, y_m)$ does not contain quantifiers. Suppose $a_1, a_2, \ldots, a_k$ is a full list of individual constants and free variables in F. Let's show how to decide if sequent $\to F$ is derivable.

Let's apply rule $(\to \forall)$ to formula F n times. The result is sequent $\to \exists y_1 \exists y_2 \ldots \exists y_m G(z_1, z_2, \ldots, z_n, y_1, y_2, \ldots, y_m)$, where $z_i, i \in [1, n]$ are different free variables, not equal to $a_i, i \in [1, k]$. Now the only way is to

apply rule $(\to \exists)$ m times and after that logical rules. Because the calculus is minus-normal, after applying rule $(\to \exists)$ variables $y_i, i \in [1, m]$ can only be replaced by an element of $\{a_1, a_2, \ldots, a_k, z_1, z_2, \ldots, z_n\}$. Thus only finite number of applications of $(\to \exists)$ is possible. Moreover, it is obvious that logical rules can also be applied only finite number of times. Thus, only finite number of derivation search trees may be obtained. If at least one of the trees is a derivation of $\to F$, then $\vDash F$, otherwise, $\nvDash F$. □

2.6 Semantic tableaux method

Suppose, sequent $F_1, F_2, \ldots, F_n \to G_1, G_2, \ldots, G_m$ is derivable in sequent calculus for predicate logic GPR and the derivation is $\mathcal{D}$. Then sequent $F_1, F_2, \ldots, F_n, \neg G_1, \neg G_2, \ldots, \neg G_m \to$ is also derivable and the derivation is:

$$\dfrac{\dfrac{\dfrac{\begin{array}{c}\mathcal{D}\\ F_1, F_2, \ldots, F_n \to G_1, G_2, \ldots, G_m\end{array}}{F_1, F_2, \ldots, F_n, \neg G_1 \to G_2, G_3, \ldots, G_m}(\neg \to)}{F_1, F_2, \ldots, F_n, \neg G_1, \neg G_2 \to G_3, G_4, \ldots, G_m}(\neg \to)}{\begin{array}{c}\ldots\\ F_1, F_2, \ldots, F_n, \neg G_1, \neg G_2, \ldots, \neg G_m \to\end{array}}$$

This technique can be used to move all the formulas from succedent to antecedent in any sequent. This eliminates the need to maintain two separate lists of formulas, because only the antecedent is left. Thus the separator symbol $\to$ can also be omitted.

It is obvious, that in a sequent calculus for such sequents the axiom must be $F, \neg F, \Gamma \to$ [1]. Moreover, the rules must be modified. Let's take GPR as a base calculus. Rules with main formulas in succedent ($(\to \neg)$, $(\to \wedge)$,...) can be removed. Rules in which the main and side formulas are in antecedent ($(\wedge \to)$, $(\vee \to)$,...) can be modified slightly to remove the antecedent part. Now only two more rules require further modification. Rule $(\neg \to)$ can be replaced by several rules, depending on the form of the main formula:

$$\dfrac{F, \Gamma \to}{\neg\neg F, \Gamma \to} \qquad \dfrac{\neg F, \Gamma \to \qquad \neg G, \Gamma \to}{\neg(F \wedge G), \Gamma \to} \qquad \dfrac{\neg F, \neg G, \Gamma \to}{\neg(F \vee G), \Gamma \to}$$

$$\dfrac{F, \neg G, \Gamma \to}{\neg(F \supset G), \Gamma \to} \qquad \dfrac{\neg F(t), \neg\exists x F(x), \Gamma \to}{\neg\exists x F(x), \Gamma \to} \qquad \dfrac{\neg F(z), \Gamma \to}{\neg\forall x F(x), \Gamma \to}$$

The change to rule $(\supset \to)$ is obvious:

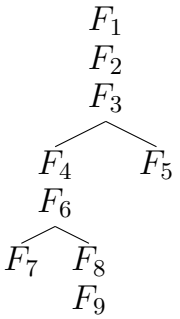
$$\dfrac{\neg F, \Gamma \to \qquad G, \Gamma \to}{F \supset G, \Gamma \to}(\supset \to)$$

[1] Although $\to$ is not necessary, it is kept for clarity: to stress that all the formulas are in the antecedent of the sequent.

The soundness and completeness of such calculus can easily be proved by showing the equivalence to *GPR*, which follows immediately from the way the calculus was formed.

Similar idea was used to create *tableaux* method (or calculus). After assessing all the advantages and disadvantages of sequent calculus, this method was described in 1972 by American scientist M. Fitting. Later, in 1994, the method was improved by F. Massacci.

A derivation search tree in tableaux calculus is displayed as oriented top-down tree:



Formula F_1 is the root, formulas F_5, F_7 and F_9 are leaves. No more than two edges exit one node and when only one edge exits the node, the edge is omitted. The direction of the edge is also omitted, because it is always the same: top-down. A path from root to some leave is called a *branch*.

A derivation search tree of formula F is a tree with $\neg F$ as a root. Every other node is obtained by applying some rule of the calculus. The rules are presented in the following form:

$$\frac{F}{\Gamma}$$

Here F is a formula, called *premise* and Γ is a *conclusion*. The rule can be applied, if the premiss is already present in the branch, which is analysed. The conclusion may consist of one or two formulas that after the application of the rule are included into the derivation search tree. If the two formulas in the conclusion of the rule are separated by $\mid$, then the derivation search tree branches.

It is said that branch is *closed*, if it contains both some formula F and its negation $\neg F$. Otherwise it is open. A tree is closed, if every branch is closed. A closed tree with root $\neg F$ is a *derivation tree* (or simply *derivation*) of formula F in tableaux calculus.

Definition 2.6.1. Tableaux calculus for predicate logic (denoted *TPR*) consists of the following rules:

Conjunction rules:

$$\frac{F \wedge G}{\begin{array}{c}F\\G\end{array}} \qquad \frac{\neg(F \vee G)}{\begin{array}{c}\neg F\\\neg G\end{array}} \qquad \frac{\neg(F \supset G)}{\begin{array}{c}F\\\neg G\end{array}}$$

Disjunction rules:

$$\frac{\neg(F \wedge G)}{\neg F \mid \neg G} \qquad \frac{F \vee G}{F \mid G} \qquad \frac{F \supset G}{\neg F \mid G}$$

Double negation rule:

$$\frac{\neg\neg F}{F}$$

Quantifier rules:

$$\frac{\forall x F(x)}{F(t)} \qquad \frac{\neg\forall x F(x)}{\neg F(z)} \qquad \frac{\exists x F(x)}{F(z)} \qquad \frac{\neg\exists x F(x)}{\neg F(t)}$$

Here z is a new free variable, which is not part of any formula of the branch, and t is some main term.

Example 2.6.2. Let's derive formula $\exists x\forall y P(x,y) \supset \forall y\exists x P(x,y)$ in tableaux calculus *TPR*. Compare this derivation to the derivation in calculus *GPR* of sequent $\rightarrow \exists x\forall y P(x,y) \supset \forall y\exists x P(x,y)$ presented in Example 2.5.5.

$$\begin{gathered}
F_1 = \neg\big(\exists x\forall y P(x,y) \supset \forall y\exists x P(x,y)\big) \\
F_2 = \exists x\forall y P(x,y) \\
F_3 = \neg\forall y\exists x P(x,y) \\
F_4 = \forall y P(x_1,y) \\
F_5 = \neg\exists x P(x,y_1) \\
F_6 = P(x_1,y_1) \\
F_7 = \neg P(x_1,y_1)
\end{gathered}$$

Here F_1 is the root — a negation of the formula, which must be derived. Formulas F_2 and F_3 are obtained from F_1 using conjunction rule. Then only quantifier rules are applied. F_4 is obtained from F_2, F_5 from F_3, F_6 from F_4 and F_7 from F_5.

Example 2.6.3. Let's derive formula $\neg(p \wedge q) \supset (\neg p \vee \neg q)$.

$$\begin{array}{c}
\neg\big(\neg(p \wedge q) \supset (\neg p \vee \neg q)\big) \\
\neg(p \wedge q) \\
\neg(\neg p \vee \neg q) \\
\diagup\quad\diagdown \\
\neg p \qquad \neg q \\
\neg\neg p \qquad \neg\neg p \\
\neg\neg q \qquad \neg\neg q \\
p \qquad\quad q
\end{array}$$

For comparison with derivations in sequent calculus several other derivations using tableaux method are provided.

Example 2.6.4. In Example 2.5.6 derivation in calculus *GPR* of formula $\Big(\neg\forall xP(x) \supset \exists x\neg P(x)\Big) \wedge \Big(\exists x\neg P(x) \supset \neg\forall xP(x)\Big)$ is presented. Let's derive it in *TPR*.

$$\neg\Big(\Big(\neg\forall xP(x) \supset \exists x\neg P(x)\Big) \wedge \Big(\exists x\neg P(x) \supset \neg\forall xP(x)\Big)\Big)$$

$$\begin{array}{cc} \neg\Big(\neg\forall xP(x) \supset \exists x\neg P(x)\Big) & \neg\Big(\exists x\neg P(x) \supset \neg\forall xP(x)\Big) \\ \neg\forall xP(x) & \exists x\neg P(x) \\ \neg\exists x\neg P(x)\Big) & \neg\neg\forall xP(x) \\ \neg P(z) & \neg P(z) \\ \neg\neg P(z) & \forall xP(x) \\ P(z) & P(z) \end{array}$$

Example 2.6.5. Finally let's use tableaux calculus *TPR* to derive formula $(F \supset G) \supset \Big((F \supset H) \supset \Big(F \supset (G \wedge H)\Big)\Big)$, which was derived in sequent calculus in Example 1.3.19. This derivation demonstrates a derivation tree with more branches.

$$F_1 = \neg\Big((F \supset G) \supset \Big((F \supset H) \supset \Big(F \supset (G \wedge H)\Big)\Big)\Big)$$

$$F_2 = F \supset G$$

$$F_3 = \neg\Big((F \supset H) \supset \Big(F \supset (G \wedge H)\Big)\Big)$$

$$F_4 = F \supset H$$

$$F_5 = \neg\Big(F \supset (G \wedge H)\Big)$$

$$F_6 = F$$

$$F_7 = \neg(G \wedge H)$$

$$F_8 = \neg F \qquad F_9 = G$$

$$F_{10} = \neg F \qquad F_{11} = H$$

$$F_{12} = \neg G \quad F_{13} = \neg H$$

The order in which the rules are applied is not important. In this derivation conjunction rules are preferred to disjunction rules, because application of disjunction rule branches the derivation tree. Thus formulas F_2 and F_3 are obtained from F_1, formulas F_4 and F_5 — from F_3 and formulas F_6 and F_7 — from F_5 by applying conjunction rules. However formulas F_8 and F_9 are obtained from F_2, formulas F_{10} and F_{11} — from F_4 and formulas F_{12} and F_{13} — from F_7 using disjunction rules. All the branches are closed: starting from the left, F_6 and F_8 closes the first branch, formulas F_6 and F_{10} — the second one, the third one is closed by formulas F_9 and F_{12} and the last one by F_{11} and F_{13}.

2.7 Compactness

In this section only propositional formulas are analysed. However, the results presented here are needed for the next section.

Definition 2.7.1. A set of formulas is *finite satisfiable*, if every finite subset of the set is satisfiable. A set of formulas $\{F_1, F_2, \dots\}$ is *satisfiable* if there exists an interpretation ν such that $\nu \vDash F_i$ for every $i \in [1, \infty)$.

If a set of formulas is not satisfiable, then it is called *contradictory*. I.e. a set of propositional formulas $\mathcal{F}$ is contradictory if for any interpretation ν there is a formula $F \in \mathcal{F}$ such that $\nu \nvDash F$.

Definition 2.7.2. A set of formulas $\mathcal{F}$ is *maximal*, if:

- $\mathcal{F}$ is finite satisfiable and
- for any formula F either $F \in \mathcal{F}$ or $\neg F \in \mathcal{F}$.

Theorem 2.7.3. *There is a bijection between the set of interpretations and the set of maximal sets of formulas.*

Proof. For every interpretation ν let's assign a set of propositional formulas $\mathcal{F}_\nu = \{F : \nu \vDash F\}$. I.e. $\mathcal{F}_\nu$ consists of formulas, which are true with interpretation ν. It is a maximal set, because interpretation ν satisfies every formula (and therefore every finite subset) of $\mathcal{F}_\nu$ and for any formula F either $\nu \vDash F$ and therefore $F \in \mathcal{F}_\nu$ or $\nu \nvDash F$, thus $\nu \vDash \neg F$ and therefore $\neg F \in \mathcal{F}_\nu$. It is obvious that for two different interpretations ν_1 and ν_2 the sets of formulas $\mathcal{F}_{\nu_1}$ and $\mathcal{F}_{\nu_2}$ are different. In fact for two interpretations to be different they must disagree on at least one propositional variable. Let $\nu_1(p) \neq \nu_2(p)$. Suppose that $\nu_1(p) = \top$ and $\nu_2(p) = \bot$ (the other case is analogous). Now formula $p \in \mathcal{F}_{\nu_1}$, however $p \notin \mathcal{F}_{\nu_2}$ and therefore $\mathcal{F}_{\nu_1} \neq \mathcal{F}_{\nu_2}$. This proves that the described correspondence (or function $f(\nu) = \mathcal{F}_\nu$) is injection.

To show that it is also a bijection let's find the inverse correspondence to the one, described in the previous paragraph. For every maximal (and thus finite satisfiable) set of formulas $\mathcal{F}$, let's assign interpretation $\nu_\mathcal{F}$ such that for any propositional variable p it would be true that $\nu_\mathcal{F}(p) = \top$ iff $p \in \mathcal{F}$. Now let's show, that for any formula F, it is true that $\nu_\mathcal{F} \vDash F$ iff $F \in \mathcal{F}$. Induction on the length of the formula is used.

Suppose $\mathrm{l}(F) = 0$, then F is a propositional variable. According to the way $\nu_\mathcal{F}$ is defined, $\nu_\mathcal{F} \vDash F$ iff $F \in \mathcal{F}$.

Suppose the assumption (that $\nu_\mathcal{F} \vDash F$ iff $F \in \mathcal{F}$) holds if $\mathrm{l}(F) < m$ (this statement is called *induction hypothesis*). Let $\mathrm{l}(F) = m$. Then different forms of formula F must be analysed. Say $F = \neg G$. Then, $\mathrm{l}(G) = m - 1$ and according to the induction hypothesis $\nu_\mathcal{F} \vDash G$ iff $G \in \mathcal{F}$. Suppose, $F = \neg G \in \mathcal{F}$, then $G \notin \mathcal{F}$, because $\mathcal{F}$ is finite satisfiable and set $\{G, \neg G\}$ is not. From this and the induction hypothesis it follows that $\nu_\mathcal{F} \nvDash G$ and therefore $\nu_\mathcal{F} \vDash F = \neg G$. If $F = \neg G \notin \mathcal{F}$, then because $\mathcal{F}$ is a maximal set $G \in \mathcal{F}$. Therefore $\nu_\mathcal{F} \vDash G$ and $\nu_\mathcal{F} \nvDash F = \neg G$.

Suppose $F = G \wedge H$. Then $\mathrm{l}(G) < m$ and $\mathrm{l}(H) < m$, thus the induction hypothesis holds for G and H. Suppose $F = G \wedge H \in \mathcal{F}$, then $\neg G \notin \mathcal{F}$ and $\neg H \notin \mathcal{F}$. This is because $\mathcal{F}$ is finite satisfiable and neither set $\{\neg G, G \wedge H\}$ nor $\{\neg H, G \wedge H\}$ is satisfiable. Now because $\mathcal{F}$ is maximal, $G \in \mathcal{F}$ and $H \in \mathcal{F}$. From the induction hypothesis it follows that $\nu_{\mathcal{F}} \vDash G$ and $\nu_{\mathcal{F}} \vDash H$ and therefore $\nu_{\mathcal{F}} \vDash F = G \wedge H$. If $G \wedge H \notin \mathcal{F}$, then because $\mathcal{F}$ is a maximal set $\neg(G \wedge H) \in \mathcal{F}$. Now $\mathcal{F}$ is finite satisfiable and set $\{\neg(G \wedge H), G, H\}$ is not, therefore $G \notin \mathcal{F}$ or $H \notin \mathcal{F}$. Because of that and according to the induction hypothesis $\nu_{\mathcal{F}} \nvDash G$ or $\nu_{\mathcal{F}} \nvDash H$ and thus $\nu_{\mathcal{F}} \nvDash G \wedge H$.

The cases, when $F = G \vee H$, $F = G \supset H$ and $F = G \leftrightarrow H$ are analysed similarly.

It is not hard to argue that $\mathcal{F}_{\nu_{\mathcal{F}}} = \mathcal{F}$ and $\nu_{\mathcal{F}_\nu} = \nu$. Thus both defined correspondences are bijections. □

The following corollary follows directly from the proof of this theorem:

Corollary 2.7.4. *For every maximal set of formulas $\mathcal{F}$ there is an interpretation ν such that for every $F \in \mathcal{F}$ it is true that $\nu \vDash F$.*

Theorem 2.7.5 (Compactness)**.** *A set of formulas is satisfiable iff it is finite satisfiable.*

Proof. If a set is satisfiable, then obviously it is finite satisfiable. That is, if there is an interpretation ν with which every formula of a set is true, then every formula of any finite subset is also true with ν.

Let's show that every finite satisfiable set $\mathcal{T}$ is satisfiable. It should be noted that not every finite satisfiable set of formulas is maximal. E.g. set $\{p_1, p_1 \wedge p_2, p_1 \wedge p_2 \wedge p_3, \dots\}$ is finite satisfiable, because interpretation $\nu(p_i) = \top, i \in [1, \infty)$ satisfies every subset, however neither formula $p_1 \vee p_2$ nor its negation belong to the set, thus it is not maximal.

To prove that any finite satisfiable set $\mathcal{T}$ is also satisfiable it is enough to find a maximal set $\mathcal{F}$ such that $\mathcal{T} \subset \mathcal{F}$. According to Corollary 2.7.4 it is possible to construct interpretation $\nu_{\mathcal{F}}$ such that $F \in \mathcal{F}$ iff $\nu_{\mathcal{F}} \vDash F$. Therefore with $\nu_{\mathcal{F}}$ every formula of $\mathcal{T}$ would be true.

Let's construct such maximal set step by step. The set of all the propositional formulas is countable. Therefore, it is possible to list all the formulas in some sequence $F_1, F_2, \dots$. Each formula occurs in the sequence exactly once. Lets construct a sequence of sets in a following way: $\mathcal{T}_0 = \mathcal{T}$ and

$$\mathcal{T}_{n+1} = \begin{cases} \mathcal{T}_n \cup \{F_n\}, & \text{if it is finite satisfiable} \\ \mathcal{T}_n \cup \{\neg F_n\}, & \text{otherwise} \end{cases}$$

At least one of the sets $\mathcal{T}_n \cup \{F_n\}$ and $\mathcal{T}_n \cup \{\neg F_n\}$ is finite satisfiable, if T_n is finite satisfiable. Indeed, suppose that neither $\mathcal{T}_n \cup \{F_n\}$ nor $\mathcal{T}_n \cup \{\neg F_n\}$ is finite satisfiable, despite that $\mathcal{T}_n$ is. In this case there are finite sets $\mathcal{T}' \subset \mathcal{T}_n \cup \{F_n\}$ and $\mathcal{T}'' \subset \mathcal{T}_n \cup \{\neg F_n\}$ such that neither $\mathcal{T}'$ nor $\mathcal{T}''$ is satisfiable. However, any finite subset of $\mathcal{T}_n$ is satisfiable, therefore $\mathcal{T}' \not\subset \mathcal{T}_n$ and $\mathcal{T}'' \not\subset \mathcal{T}_n$. From this it follows that $F_n \in \mathcal{T}'$ and $\neg F_n \in \mathcal{T}''$. Let $\mathcal{T}' = \mathcal{T}'_n \cup \{F_n\}$, $\mathcal{T}'' = \mathcal{T}''_n \cup \{\neg F_n\}$. Then $\mathcal{T}'_n \subset \mathcal{T}_n$ and $\mathcal{T}''_n \subset \mathcal{T}_n$. Now

because $\mathcal{T}_n$ is finite satisfiable, there is an interpretation ν with which any formula of finite set $\mathcal{T}'_n \cup \mathcal{T}''_n \subset \mathcal{T}_n$ is true. More specifically, any formula of $\mathcal{T}'_n$ and $\mathcal{T}''_n$ is true with ν. Finally, either $\nu \vDash F_n$ or $\nu \vDash \neg F_n$, and therefore either $\mathcal{T}'$ or $\mathcal{T}''$ is finite satisfiable. Thus the contradiction is obtained.

Now let $\mathcal{F} = \bigcup_{i=0}^{\infty} \mathcal{T}_n$. It is true that $\mathcal{T} \subset \mathcal{F}$, because $\mathcal{T} = \mathcal{T}_0$. Moreover from the construction of $\mathcal{F}$ it follows that it is a maximal set. Therefore, according to Corollary 2.7.4 there is an interpretation ν with which any formula of $\mathcal{F}$ (and also those of $\mathcal{T}$) is true. This proves that $\mathcal{T}$ is a satisfiable set. □

Corollary 2.7.6. *If some (infinite) set of formulas is contradictory, then there is a finite contradictory subset of the set.*

Proof. Indeed say that set of formulas $\mathcal{F}$ is contradictory. Then it is not satisfiable. According to Theorem 2.7.5 a set is finite satisfiable, only if it is satisfiable. Thus $\mathcal{F}$ is not finite satisfiable. Now according to Definition 2.7.5 there is a finite subset of $\mathcal{F}$, which is not satisfiable and thus contradictory. □

2.8 Semantic trees

Now let's return to predicate logic. In this section predicate formulas with function symbols but without free variables are analysed. Moreover, formulas must be transformed into normal prenex form and skolemized.

Definition 2.8.1. A *Herbrand universe* $\mathcal{H}$ of formula F is defined as follows:

- Every constant of formula F is part of $\mathcal{H}$. If F does not contain any constants, then $a \in \mathcal{H}$.
- If f^n is n-place function symbol in formula F and $t_1, t_2, \ldots, t_n \in \mathcal{H}$, then $f(t_1, t_2, \ldots, t_n) \in \mathcal{H}$

It must be noted that according to the definition Herbrand universe of any formula is never empty. It is finite, if the formula does not contain function symbols, or infinite but countable.

Definition 2.8.2. A *Herbrand base* $\mathcal{B}$ of formula F is the set of all the atomic formulas $P(t_1, t_2, \ldots, t_n)$, where P^n is n-place predicate variable of F and $t_i, i \in [1, n]$ are some elements of Herbrand universe of F.

Definition 2.8.3. *H-interpretation* (or *Herbrand interpretation*) of formula F is a set $\{\alpha_1 P_1, \alpha_2 P_2, \ldots\}$, where $\alpha_i \in \{\neg, \varnothing\}$ and $P_i \in \mathcal{B}, i \in [1, \infty)$. If $\alpha_i = \neg$, then it is understood that P_i is false with the H-interpretation, otherwise P_i is true.

French logician J. Herbrand in 1930 proved the following theorem:

Theorem 2.8.4. *A formula F is satisfiable iff it is satisfiable in Herbrand universe of F.*

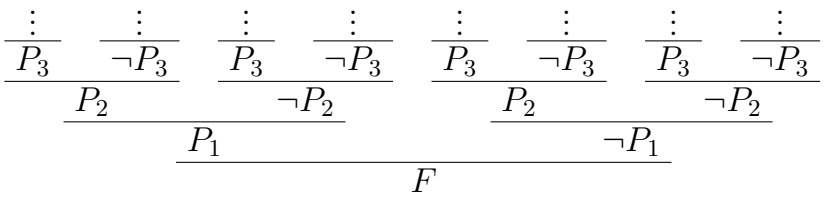
This means that to check if formula is satisfiable there is no need to go through all the possible interpretations. It is enough to go through all the possible values of predicates in Herbrand base (H-interpretations). However, Herbrand base is usually infinite, although if the formula does not contain function variables, it is finite. Recall, that the formula must be skolemized. Thus Herbrand base of the formula is finite if in the normal prenex form quantifier $\exists$ does not occur in the scope of quantifier $\forall$ and, obviously, the normal prenex form also doesn't contain function symbols.

If formula F is true with some H-interpretation, then the H-interpretation is called the *H-model* (or *Herbrand model*) of the formula.

It must be stressed, that Theorem 2.8.4 applies only to skolemized normal prenex form of the formulas. E.g. formula $P(a) \wedge \exists x \neg P(x)$ is satisfiable, however it does not have H-model, because it is not skolemized. Indeed, let $\mathcal{M} = \{a, b\}$, let's define P in a following way $P(a) = \top$ and $P(b) = \bot$. With this interpretation formula is satisfiable. However, $\mathcal{H} = \{a\}$ and the only possible H-interpretations are $\{P(a)\}$ and $\{\neg P(a)\}$. In both cases the formula is false, therefore, the formula does not have a H-model.

Therefore formulas of the form $\forall x_1 \forall x_2 \ldots \forall x_n G(x_1, x_2, \ldots, x_n)$, where $G(x_1, x_2, \ldots, x_n)$ does not contain any quantifiers, are analysed. Semantic tree is used to check if formula is satisfiable (or rather if it is not satisfiable). According to the theorem it is enough to analyse only predicates which are defined in Herbrand base $\mathcal{H}$ of the formula. That is, the values of the individual variables are from the set $\mathcal{H}$. Thus a set of formulas without quantifiers $\mathcal{T} = \{G(t_1, t_2, \ldots, t_n) : t_i \in \mathcal{H}, i \in [1, n]\}$ can be inspected. The formula is satisfiable iff set $\mathcal{T}$ is satisfiable. According to Corollary 2.7.6 of the compactness theorem, $\mathcal{T}$ is not satisfiable (is contradictory), iff there is a finite contradictory subset of $\mathcal{T}$. Let's denote the subset $\mathcal{T}'$. Such subset contains only finite number of atomic formulas.

Let's present all the H-interpretations of formula F as the tree. For that purpose let's list the members of Herbrand base $\mathcal{B}$ of F in some order. Suppose the order is $\{P_1, P_2, P_3, \ldots\}$, then the *semantic tree* of F is a tree of the form:



Every branch is infinite, if $\mathcal{B}$ is infinite (if F contains function variables), and every branch corresponds to some H-interpretation. Now let $\mathcal{T}_i$ be a set of formulas of $\mathcal{T}$, which contain only atomic formulas $P_j, j \in [1, i]$. First, let's check if $\mathcal{T}_1$ is contradictory, then let's continue with $\mathcal{T}_2$, $\mathcal{T}_3$, etc.... Recall that if formula F is contradictory, then there is a finite contradictory subset $\mathcal{T}' \subset \mathcal{T}$ with only finite number of atomic formulas. Let all the atomic formulas of $\mathcal{T}'$ be $P_{i_1}, P_{i_2}, \ldots, P_{i_m}$ such that $i_j < i_{j+1}, j \in [1, m)$.

Thus $\mathcal{T}' \subseteq \mathcal{T}_{i_m}$. However $\mathcal{T}'$ is contradictory thus any $\mathcal{T}'' \supseteq \mathcal{T}'$ is also contradictory and obviously, $\mathcal{T}_{i_m}$ is contradictory.

To sum up, if F is contradictory, then $\mathcal{T}_{i_m}$ is contradictory and it will be encountered while checking all the sets starting with $\mathcal{T}_1$. However, if F is satisfiable, such set will not be found and the process will continue infinitely.

Similar approach is applied in semantic trees. A finite subbranch representing interpretation $\{\alpha_1 P_1, \alpha_2 P_2, \dots, \alpha_k P_k\}, \alpha_i \in \{\neg, \varnothing\}, i \in [1, k]$ is chosen. It is checked if at least one formula of $\mathcal{T}_k$ is false with the interpretation. If such formula exists (let's denote it F'), then the subtree in the branch above $\alpha_k P_k$ is removed and replaced by symbol $\oplus$. Of course, at first interpretations in which $k = 1$ are checked, followed by the ones in which $k = 2$ etc.... If F is contradictory, then in such way infinite semantic tree can be transformed into finite one. At worst every interpretation in which $k = i_m$ must be checked, however usually the process in most of the branches terminates much quicker. The contradictory subset $\mathcal{T}'$ consists of formulas obtained analogously to F'. If however F is satisfiable, then this process continues infinitely.

A semantic tree is analogous to the truth table method in propositional logic.

Example 2.8.5. Let's start by finding a finite semantic tree of predicate formula $\forall x\Big(P\big(x, f(a)\big) \wedge \neg P\big(x, x\big)\Big)$.

$$\mathcal{H} = \{a, f(a), f\big(f(a)\big), \dots\}, \mathcal{B} = \{P\big(a, a\big), P\big(a, f(a)\big), P\big(f(a), f(a)\big), \dots\}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\oplus}{P\big(f(a), f(a)\big)} \quad \dfrac{\oplus}{\neg P\big(f(a), f(a)\big)}}{P\big(a, f(a)\big)} \quad \dfrac{\oplus}{\neg P\big(a, f(a)\big)}}{\neg P(a, a)} \quad \dfrac{\oplus}{P(a, a)}}{\forall x\Big(P\big(x, f(a)\big) \wedge \neg P\big(x, x\big)\Big)}$$

In this case set $\mathcal{T}$ is equal to:

$$\{F_1 = P\big(a, f(a)\big) \wedge \neg P\big(a, a\big),$$

$$F_2 = P\big(f(a), f(a)\big) \wedge \neg P\big(f(a), f(a)\big),$$

$$F_3 = P\Big(f\big(f(a)\big), f(a)\Big) \wedge \neg P\Big(f\big(f(a)\big), f\big(f(a)\big)\Big), \dots\}$$

Let's analyse branches from left to right. With interpretation $\{P(a, a)\}$ formula F_1 is false. With interpretations $\{\neg P\big(a, a\big), P\big(a, f(a)\big), P\big(f(a), f(a)\big)\}$ and $\{\neg P\big(a, a\big), P\big(a, f(a)\big), \neg P\big(f(a), f(a)\big)\}$ formula F_2 is false. With interpretation $\{\neg P\big(a, a\big), \neg P\big(a, f(a)\big)\}$ once again formula F_1 is false. Thus the contradictory subset consists of two formulas: F_1 and F_2.

Example 2.8.6. Now let's find a finite semantic tree of another formula: $\forall x\Big(P\big(x\big) \wedge \big(\neg P\big(x\big) \vee Q\big(f(x)\big)\big) \wedge \neg Q\big(f(a)\big)\Big)$.

$\mathcal{H} = \{a, f(a), f\big(f(a)\big), \dots\}$, $\mathcal{B} = \{P\big(a\big), Q\big(a\big), P\big(f(a)\big), Q\big(f(a)\big), \dots\}$.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\oplus}{Q\big(f(a)\big)} \quad \dfrac{\oplus}{\neg Q\big(f(a)\big)}}{P\big(f(a)\big)} \quad \dfrac{\oplus}{\neg P\big(f(a)\big)}}{Q(a)} \quad \dfrac{\dfrac{\dfrac{\oplus}{Q\big(f(a)\big)} \quad \dfrac{\oplus}{\neg Q\big(f(a)\big)}}{P\big(f(a)\big)} \quad \dfrac{\oplus}{\neg P\big(f(a)\big)}}{\neg Q(a)}}{P(a)} \quad \dfrac{\oplus}{\neg P(a)}}{\forall x\Big(P\big(x\big) \wedge \big(\neg P\big(x\big) \vee Q\big(f(x)\big)\big) \wedge \neg Q\big(f(a)\big)\Big)}$$

Using the knowledge obtained while making the tree, it is possible to produce simpler semantic tree. For that purpose let's reorder the elements of $\mathcal{B}$ as follows: $\{P\big(a\big), Q\big(f(a)\big), \dots\}$. The simplified tree is:

$$\dfrac{\dfrac{\dfrac{\oplus}{Q(f(a))} \quad \dfrac{\oplus}{\neg Q(f(a))}}{P(a)} \quad \dfrac{\oplus}{\neg P(a)}}{\forall x\Big(P\big(x\big) \wedge \big(\neg P\big(x\big) \vee Q\big(f(x)\big)\big) \wedge \neg Q\big(f(a)\big)\Big)}$$

Similar simplifications can be applied to the previous example.

2.9 Resolution method

2.9.1 The description of resolution calculus

Once again, in this section only skolemized normal prenex form predicate formulas are analysed. In order to apply resolution method a set of disjuncts must be obtained. As mentioned in the previous section, all the analysed formulas are of the form $\forall x_1 \forall x_2 \dots \forall x_n G(x_1, x_2, \dots, x_n)$. This formula is not satisfiable iff a set $\mathcal{T} = \{G(t_1, t_2 \dots, t_n) : t_i \in \mathcal{H}, i \in [1, n]\}$, where $\mathcal{H}$ is Herbrand universe of the formula, is contradictory. From the fact that $\mathcal{H}$ is either finite or countable and there are only finite number (n) of individual variables in the formula it follows that $\mathcal{T}$ is either finite or countable.

Now let's transform formula $G(x_1, x_2, \dots, x_n)$ into NCF. Suppose that NCF of $G(x_1, x_2, \dots, x_n)$ is $\bigwedge\limits_{j=1}^{m} H_j(x_1, x_2, \dots, x_n)$, where $H_j, j \in [1, m]$ are disjuncts. Let's analyse a set of disjuncts

$$\mathcal{S} = \{H_j(t_1, t_2, \dots, t_n) : t_i \in \mathcal{H}, i \in [1, n], j \in [1, m]\}$$

It is clear that set $\mathcal{S}$ is contradictory iff set $\mathcal{T}$ is contradictory. Moreover, from formula $G(x_1, x_2, \dots, x_n)$ only finite number of disjuncts can be produced, therefore set $\mathcal{S}$ is also either finite or countable. This means that it

is possible to list every element of $\mathcal{S}$ in some order. Let $\mathcal{S} = \{H'_1, H'_2, \dots\}$. Formulas of $\mathcal{S}$ contain neither individual variables nor quantifiers, thus it is not hard to apply propositional resolution rule to such formulas.

The most straight forward way of applying resolution method in this case would be to check if set of disjuncts $\{H'_1, H'_2\}$ is derivable in resolution calculus. If it is derivable, then the formula is not satisfiable. Otherwise, set $\{H'_1, H'_2, H'_3\}$ must be checked. The compactness Theorem 2.7.5 and the resolution method ensures that if the formula is not satisfiable, then there is some k for which set $\{H'_1, H'_2, \dots, H'_k\}$ is derivable in resolution calculus. If the formula is satisfiable, then this process is infinite.

However this procedure is not very convenient. In 1965 American logician J.A. Robinson described another way to apply resolution method to predicate formulas. Analogous method was also described by Russian S.J. Maslov one year earlier and called *reverse method*. In order to present the method several definitions are needed.

Definition 2.9.1. A substitution α is called a *unifier* of formulas (terms) F_1 and F_2, if $F_1\alpha = F_2\alpha$.

Definition 2.9.2. A unifier α is the *most general* one for formulas (terms) F_1 and F_2, if for any unifier β of F_1 and F_2 there is a substitution γ, such that $\beta = \alpha \circ \gamma$ (i.e. β is a composition of α and γ).

Substitution α is a *unifier* of set $\{F_1, F_2, \dots, F_n\}$, if $F_1\alpha = F_2\alpha = \dots = F_n\alpha$. Substitution α is the *most general unifier* of the set, if it is a unifier of the set and for any unifier of the set β there is a substitution γ such that $\beta = \alpha \circ \gamma$.

Example 2.9.3. A substitution $\{c/x, d/y, f(d)/z\}$ is a unifier of formulas $P\big(g(y,c), z\big)$ and $P\big(g(d,x), f(d)\big)$.

Two formulas (terms) F and G are *unifiable* if there is a unifier of F and G.

Example 2.9.4. Atomic formulas $P\Big(x, f\big(f(y)\big)\Big)$ and $P\Big(f(z), f\Big(f\big(g(a)\big)\Big)\Big)$ are unifiable. The unifiers are, for example, $\Big\{f(a)/x, g(a)/y, a/z\Big\}$ and $\Big\{f\big(f(a)\big)/x, g(a)/y, f(a)/z\Big\}$. The most general unifier is $\Big\{f(z)/x, g(a)/y\Big\}$.

Of course, not all the expressions are unifiable. E.g. formulas $F(t_1)$ and $F(t_2)$ are not unifiable, if terms t_1 and t_2 are:

- two different constants,
- a variable x and term t which contains x, $t \neq x$,
- a constant and function symbol,
- terms starting with different function symbols.

Now a rule of resolution for predicate formulas is:

$$\frac{F \vee l_1 \qquad l_2 \vee G}{(F \vee G)\alpha}\,\alpha$$

Here l_1 and l_2 are literals, one of which is $P(t_1^1, t_2^1, \ldots, t_n^1)$ and another one is $\neg P(t_1^2, t_2^2, \ldots, t_n^2)$. Substitution α is a unifier of $P(t_1^1, t_2^1, \ldots, t_n^1)$ and $P(t_1^2, t_2^2, \ldots, t_n^2)$.

In order to derive formula in resolution calculus the following steps must be taken:

1. it must be transformed into normal prenex form;
2. it must be skolemized, $\forall$ quantifiers must be removed;
3. NCF of skolemized formula must be found;
4. a set composed of the disjuncts of the NCF must be created;
5. the set of disjuncts must be derived in resolution calculus using resolution rule for predicate logic.

Other properties of resolution calculus for predicate logic are the same as the ones of resolution calculus for propositional logic. So let's continue with an example.

Example 2.9.5. Using resolution method let's prove that if some binary relation is irreflexive and transitive, then it is asymmetric.

Let's denote a binary relation between x and y as predicate $P(x, y)$. Irreflexivity is defined as $F_1 = \forall x \neg P(x, x)$ and transitivity is defined as $F_2 = \forall x \forall y \forall z \Big(\big(P(x, y) \wedge P(y, z) \big) \supset P(x, z) \Big)$. Asymmetry is defined as $F_3 = \forall x \forall y \big(P(x, y) \supset \neg P(y, x) \big)$.

The aim is to prove that from F_1 and F_2 follows F_3. This can be done by showing that set $\{F_1, F_2, \neg F_3\}$ is contradictory.

Let's transform the set into set of disjuncts. After removing $\forall$ quantifier from F_1, a disjunct $\neg P(x, x)$ is obtained. Formula F_2 is also skolemized, thus let's remove all the $\forall$ quantifiers and transform it into NCF:

$$\big(P(x, y) \wedge P(y, z) \big) \supset P(x, z) = \neg \big(P(x, y) \wedge P(y, z) \big) \vee P(x, z) =$$

$$= \neg P(x, y) \vee \neg P(y, z) \vee P(x, z)$$

Thus formula F_2 results in one disjunct: $\neg P(x, y) \vee \neg P(y, z) \vee P(x, z)$. Finally with F_3 every step of the algorithm must be performed. First it must be transformed into normal prenex form:

$$\neg \forall x \forall y \big(P(x, y) \supset \neg P(y, x) \big) = \exists x \exists y \neg \big(P(x, y) \supset \neg P(y, x) \big) =$$

$$= \exists x \exists y \neg \big(\neg P(x, y) \vee \neg P(y, x) \big) = \exists x \exists y \big(P(x, y) \wedge P(y, x) \big)$$

After skolemization formula $P(a,b) \wedge P(b,a)$ is obtained and this results in two disjuncts $P(a,b)$ and $P(b,a)$.

Therefore, $\mathcal{S} = \{\neg P(x,x), \neg P(x,y) \vee \neg P(y,z) \vee P(x,z), P(a,b), P(b,a)\}$. Now the following derivation can be obtained:

$$\dfrac{\dfrac{\dfrac{P(a,b) \qquad \neg P(x,y) \vee \neg P(y,z) \vee P(x,z)}{\neg P(b,z) \vee P(a,z)}\,\{{}^{a}/_{x},\,{}^{b}/_{y}\} \qquad P(b,a)}{P(a,a)}\,\{{}^{a}/_{z}\} \qquad \neg P(x,x)}{\square}\,\{{}^{a}/_{x}\}$$

Sometimes it is not clear how to start or proceed the derivation search in resolution calculus. To help with this task, it is possible to add additional requirements to the premisses or conclusion of application of resolution rule. Such requirements are called *tactics*. They usually allow to lessen the number of possible applications of the rule. It is said that tactics is *complete*, if empty disjunct is derivable iff it is derivable using the tactics.

Let's describe several complete tactics.

2.9.2 Linear tactics

Suppose formula F is derivable from set $\mathcal{S}$ and $T_1, T_2, \ldots T_n$ is a sequence of applications of resolution rule in the derivation. If a conclusion of T_n is F and for every $i \in [2,n]$ one of the premisses of T_i is a conclusion of T_{i-1}, then it is said that the derivation follows the linear tactics.

Example 2.9.6. Let's transform the following derivation into the one that follows the linear tactics:

$$\dfrac{\dfrac{\dfrac{\mathcal{D}}{F_3 \vee p} \qquad \dfrac{F_1 \vee \neg p \vee q \qquad F_2 \vee \neg q}{F_1 \vee F_2 \vee \neg p}}{F_1 \vee F_2 \vee F_3}$$

Here $\mathcal{D}$ denotes a derivation of $F_3 \vee p$, which follows linear tactics.

The solution is as follows:

$$\dfrac{\dfrac{\dfrac{\dfrac{\mathcal{D}}{F_3 \vee p} \qquad F_1 \vee \neg p \vee q}{F_1 \vee F_3 \vee q} \qquad F_2 \vee \neg q}{F_1 \vee F_2 \vee F_3}$$

Example 2.9.7. Let's derive the set used in Example 1.3.25 in resolution calculus by following the linear tactics:

$$\dfrac{\dfrac{F \qquad \dfrac{\neg F \vee H \qquad \dfrac{\neg F \vee G \qquad \neg G \vee \neg H}{\neg F \vee \neg H}}{\neg F}}{\square}$$

Notice, that a derivation presented in Example 2.9.5 follows the linear tactics.

2.9.3 Tactics of semantic resolution

Suppose, $\mathcal{S}$ is a set of disjuncts and $\mathcal{I}$ is some interpretation, which divides $\mathcal{S}$ into two subsets $\mathcal{S}_+$ and $\mathcal{S}_-$. Here $\mathcal{S}_+$ consists of formulas from $\mathcal{S}$ that are true with interpretation $\mathcal{I}$ and $\mathcal{S}_-$ consists of formulas from $\mathcal{S}$ that are false with interpretation $\mathcal{I}$.

According to the tactics, premisses of the application of resolution rule must be part of different subsets. After the application, the conclusion is also assigned to $\mathcal{S}_+$ or $\mathcal{S}_-$ depending on if it is true with $\mathcal{I}$ or not.

Example 2.9.8. Let $\mathcal{S} = \{p \vee q \vee \neg r, \neg p \vee q, \neg q \vee \neg r, r\}$ and interpretation $\mathcal{I}$ defined as follows: $\mathcal{I}(p) = \top$, $\mathcal{I}(q) = \top$ and $\mathcal{I}(r) = \bot$.

In this case $\mathcal{S}_+ = \{p \vee q \vee \neg r, \neg p \vee q, \neg q \vee \neg r\}$ and $\mathcal{S}_- = \{r\}$.

The derivation is as follows:

$$\dfrac{\dfrac{\dfrac{p \vee q \vee \neg r \qquad r}{p \vee q} \qquad \dfrac{\neg q \vee \neg r \qquad r}{\neg q}}{p} \qquad \dfrac{\neg p \vee q \qquad \dfrac{\neg q \vee \neg r \qquad r}{\neg q}}{\neg p}}{\square}$$

It should be noted that the conclusions are included into subsets as soon as they are obtained. At the end $\mathcal{S}_+ = \{p \vee q \vee \neg r, \neg p \vee q, \neg q \vee \neg r, p \vee q, p\}$ and $\mathcal{S}_- = \{r, \neg q, \neg p\}$.

Example 2.9.9. Let's find the derivation of

$$\mathcal{S} = \{p_1 \vee p_2, p_1 \vee \neg p_2, \neg p_1 \vee p_2, \neg p_1 \vee \neg p_2\}$$

using semantic resolution tactics, when the interpretation $\mathcal{I}$ is $\mathcal{I}(p_1) = \top$, $\mathcal{I}(p_2) = \bot$.

Based on the given information $\mathcal{S}_+ = \{p_1 \vee p_2, p_1 \vee \neg p_2, \neg p_1 \vee \neg p_2\}$ and $\mathcal{S}_- = \{\neg p_1 \vee p_2\}$. Thus the derivation would be as follows:

$$\dfrac{\dfrac{p_1 \vee \neg p_2 \qquad \dfrac{p_1 \vee p_2 \qquad \neg p_1 \vee p_2}{p_2}}{p_1} \qquad \dfrac{\neg p_1 \vee \neg p_2 \qquad \neg p_1 \vee p_2}{\neg p_1}}{\square}$$

The subsets are:

$$\mathcal{S}_+ = \{p_1 \vee p_2, p_1 \vee \neg p_2, \neg p_1 \vee \neg p_2, p_1\}, \mathcal{S}_- = \{\neg p_1 \vee p_2, p_2, \neg p_1\}$$

Example 2.9.10. Let's derive the set used in Example 1.3.25 by following the semantic resolution tactics. Let $\mathcal{I}(F) = \mathcal{I}(G) = \mathcal{I}(H) = \top$. Then $\mathcal{S}_+ = \{\neg F \vee G, \neg F \vee H, F\}$ and $\mathcal{S}_- = \{\neg G \vee \neg H\}$. In that case derivation presented in Example 2.9.7 follows the semantic resolution tactics. After the derivation only $\mathcal{S}_-$ is changed. It is equal to $\{\neg G \vee \neg H, \neg F \vee \neg H, \neg F\}$.

Example 2.9.11. Let's derive the set used in Example 2.9.5 by following the semantic resolution tactics. In this case Herbrand universe $\mathcal{H} = \{a, b\}$ and Herbrand base $\mathcal{B} = \{P(a, a), P(a, b), P(b, a), P(b, b)\}$. Now let's analyse

the most primitive H-interpretation $\{P(a,a), P(a,b), P(b,a), P(b,b)\}$. It divides the set $\mathcal{S}$ into the following parts:

$$\mathcal{S}_+ = \{\neg P(x,y) \vee \neg P(y,z) \vee P(x,z), P(a,b), P(b,a)\}, \mathcal{S}_- = \{\neg P(x,x)\}$$

The derivation is:

$$\dfrac{\dfrac{P(b,a) \qquad \dfrac{P(a,b) \qquad \dfrac{\neg P(x,y) \vee \neg P(y,z) \vee P(x,z) \qquad \neg P(x,x)}{\neg P(x,y) \vee \neg P(y,x)}\{x/z\}}{\neg P(b,a)}\{a/x, b/y\}}{\square}\{\}$$

2.9.4 Absorption tactics

It is said that disjunct $F_1 = l \vee G$ is absorbed by another disjunct F_2, if $F_2 = \neg l \vee G \vee G'$. Here l is a literal, $\neg\neg l = l$, G and G' are disjuncts and G' may be empty.

According to absorption tactics, a resolution rule can be applied only if one of the disjuncts absorbs the other one. More precisely, the rule can be applied to F_1 and F_2 with unifier α is either $F_1\alpha$ absorbs or is absorbed by $F_2\alpha$.

Example 2.9.12. Suppose, it is possible to transform any derivation tree to the one which follows the absorption tactics, if there are no more than n applications of resolution rule (induction hypothesis). A derivation tree of $F \vee G$ with $n+1$ application of resolution rule is as follows:

$$\dfrac{\dfrac{\dfrac{\mathcal{D}_1}{F \vee p \vee q} \qquad \dfrac{\mathcal{D}_2}{F \vee p \vee \neg q}}{F \vee p} \qquad \dfrac{\mathcal{D}_3}{G \vee \neg p}}{F \vee G}(*)$$

The $(*)$ denotes an application of resolution rule which doesn't follow the absorption tactics. Let's prove, that it is possible to derive disjunct $F \vee G$ in resolution method using absorption tactics.

Let's analyse the two derivations:

$$\dfrac{\dfrac{\mathcal{D}_3}{G \vee \neg p} \qquad \dfrac{\mathcal{D}_1}{F \vee p \vee q}}{F \vee G \vee q} \qquad \text{and} \qquad \dfrac{\dfrac{\mathcal{D}_3}{G \vee \neg p} \qquad \dfrac{\mathcal{D}_2}{F \vee p \vee \neg q}}{F \vee G \vee \neg q}$$

Both of them have no more than n applications of resolution rule. Therefore, according to the induction hypothesis, it is possible to derive both disjuncts $F \vee G \vee q$ and $F \vee G \vee \neg q$ using absorption tactics. Let's denote the resulting derivations $\mathcal{D}_1'$ and $\mathcal{D}_2'$. Then the derivation of $F \vee G$ that follows the absorption tactics is as follows:

$$\dfrac{\dfrac{\mathcal{D}_1'}{F \vee G \vee q} \qquad \dfrac{\mathcal{D}_2'}{F \vee G \vee \neg q}}{F \vee G}$$

Notice that the derivation provided in Example 1.3.25 follows the absorption tactics as well as both derivations presented in Examples 2.9.5 and 2.9.11.

2.10 Intuitionistic logic

A lot of theorems of mathematics prove the existence of some object with some properties. E.g. Gentzen Hauptsatz Theorem 2.5.2 proves that for every predicate sequent which is derivable in GPR_o there is a derivation without applications of the cut rule. There are two ways to prove such theorem. One is simply to give an example of such object or in a case of Gentzen Hauptsatz describe a way to derive every derivable sequent without using the cut rule. Another method is to prove the existence without describing the existing object. The first type of proof is called *constructive* and the second one *non-constructive*.

The proof of Gentzen Hauptsatz provided in [5] is in fact constructive. It describes how the applications of the cut rule can be eliminated from derivations of GPR_o. However it is not always possible to find a constructive proof. In mathematics constructivity is usually lost when using the contradiction technique. Suppose, that the aim is to prove F. If proof by contradiction is performed, then hypothesis that $\neg F$ holds is made and then it is showed, that the hypothesis is wrong and thus $\neg\neg F$ is true. Therefore the conclusion is made that F is also true. However, in intuitionistic logic formulas F and $\neg\neg F$ have different meanings.

For a comparison of constructive and non-constructive proofs let's analyse the to ways to prove the following theorem

Theorem 2.10.1 (Non-constructive proof)**.** *There are two irrational numbers a and b such that a^b is a rational number.*

Proof. Suppose that $\sqrt{2}^{\sqrt{2}}$ is a rational number. Then if $a = b = \sqrt{2}$ we obtain that a and b are irrational, but a^b is rational. Otherwise, if $\sqrt{2}^{\sqrt{2}} \notin \mathbb{Q}$, let's take $a = \sqrt{2}^{\sqrt{2}}$ and $b = \sqrt{2}$. Then

$$a^b = \left(\sqrt{2}^{\sqrt{2}}\right)^{\sqrt{2}} = \sqrt{2}^{\sqrt{2}\times\sqrt{2}} = \sqrt{2}^2 = 2$$

Therefore $a^b \in \mathbb{Q}$. □

The theorem is proved, however it is not clear what is the value of a (number $b = \sqrt{2}$ in both analysed cases). However, it is possible to give a constructive proof of this theorem.

Theorem 2.10.2 (Constructive proof)**.** *There are two irrational numbers a and b such that a^b is a rational number.*

Proof. Let $a = \sqrt{2}$ and $b = \log_2 9$. It is obvious that $a \notin \mathbb{Q}$ and $b > 0$. Now if $b \in \mathbb{Q}$, then there are numbers $m, n \in \mathbb{N}\backslash\{0\}$ such that $\log_2 9 = \frac{m}{n}$. According to the definition of logarithm $2^{\frac{m}{n}} = 9$ and thus $2^m = 9^n$. However 2^m is always even and 9^n is always odd for any $m, n \in \mathbb{N}\backslash\{0\}$. Thus $b \notin \mathbb{Q}$. Now

$$a^b = \sqrt{2}^{\log_2 9} = 2^{\frac{1}{2}\log_2 9} = 2^{\log_2 9^{\frac{1}{2}}} = 2^{\log_2 \sqrt{9}} = 2^{\log_2 3} = 3$$

Obviously $3 \in \mathbb{Q}$ and thus $a^b \in \mathbb{Q}$. □

From the last proof it is obvious that two irrational numbers are $\sqrt{2}$ and $\log_2 9$. However, as mentioned earlier, it is not always possible to provide a constructive proof. In some cases it is even impossible. The following two theorems shows such example.

Theorem 2.10.3 (Bolzano–Weierstrass)**.** *In every bounded sequence there is a convergent subsequence.*

Theorem 2.10.4. *There is no algorithm, which describes how to find a convergent subsequence in any bounded sequence.*

In order to have another constructive mathematics, i.e. the one, in which by proving the existence of some objects it would be possible to find those objects using the proof, another logic is needed. It is called intuitionistic and was created in 1930 by Dutch logician A. Heyting.

Hilbert-type intuitionistic calculus for intuitionistic logics is the same as *HPR* with one difference:

Definition 2.10.5. Hilbert-type calculus for intuitionistic logic (denoted *HIN*) is obtained from calculus *HPR* by replacing axiom *4.3* ($\neg\neg A \supset A$) by $\neg A \supset (A \supset B)$.

Intuitionistic sequent calculus differs from the predicate one in one aspect: in intuitionistic calculus the succedent of the sequent can have at most one formula.

Definition 2.10.6. Gentzen-type calculus for intuitionistic logic (denoted *GIN*) is composed of axiom $F \to F$, structural rules:

Weakening:

$$\frac{\Gamma \to \Delta}{F, \Gamma \to \Delta}\,(w \to) \qquad \frac{\Gamma \to}{\Gamma \to F}\,(\to w)$$

Contraction:

$$\frac{F, F, \Gamma \to \Delta}{F, \Gamma \to \Delta}\,(c \to)$$

Exchange:

$$\frac{\Gamma_1, G, F, \Gamma_2 \to \Delta}{\Gamma_1, F, G, \Gamma_2 \to \Delta}\,(e \to)$$

Logical rules:

Negation:

$$\frac{\Gamma \to F}{\neg F, \Gamma \to \Delta}\,(\neg \to) \qquad \frac{F, \Gamma \to}{\Gamma \to \neg F}\,(\to \neg)$$

Conjunction:

$$\frac{F, G, \Gamma \to \Delta}{F \wedge G, \Gamma \to \Delta}\,(\wedge \to) \qquad \frac{\Gamma \to F \qquad \Gamma \to G}{\Gamma \to F \wedge G}\,(\to \wedge)$$

Disjunction:

$$\frac{F, \Gamma \to \Delta \qquad G, \Gamma \to \Delta}{F \vee G, \Gamma \to \Delta}\,(\vee \to)$$

$$\frac{\Gamma \to F}{\Gamma \to F \vee G}\,(\to \vee)_1 \qquad \frac{\Gamma \to G}{\Gamma \to F \vee G}\,(\to \vee)_2$$

Implication:

$$\frac{\Gamma \to F \qquad G, \Gamma \to \Delta}{F \supset G, \Gamma \to \Delta}\,(\supset \to) \qquad \frac{F, \Gamma \to G}{\Gamma \to F \supset G}\,(\to \supset)$$

Quantifier rules:

Existence:

$$\frac{F(z), \Gamma \to \Delta}{\exists x F(x), \Gamma \to \Delta}\,(\exists \to) \qquad \frac{\Gamma \to F(t)}{\Gamma \to \Delta, \exists x F(x)}\,(\to \exists)$$

Universality:

$$\frac{F(t), \forall x F(x), \Gamma \to \Delta}{\forall x F(x), \Gamma \to \Delta}\,(\forall \to) \qquad \frac{\Gamma \to F(z)}{\Gamma \to \forall x F(x)}\,(\to \forall)$$

Variable z and term t must satisfy the same requirements as in the case of GPR_o.

And the cut rule:

$$\frac{\Gamma_1 \to F \qquad F, \Gamma_2 \to \Delta}{\Gamma_1, \Gamma_2 \to \Delta}\,(\text{cut } F)$$

Note that set Δ in this calculus represents one formula or an empty set. Succedent is not necessary, however it must contain no more than one formula.

It is possible to prove that every formula which is derivable in *GIN* is also derivable without cut. However structural rules cannot be eliminated. Let's demonstrate this with an example.

Example 2.10.7. Contraction rule is necessary for calculus *GIN*. Formula $\neg\neg(p \vee \neg p)$ is derivable:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{p \to p}{p \to p \vee \neg p}(\to \vee)_1}{\neg(p \vee \neg p), p \to}(\neg \to)}{p, \neg(p \vee \neg p) \to}(e \to)}{\neg(p \vee \neg p) \to \neg p}(\to \neg)}{\neg(p \vee \neg p) \to p \vee \neg p}(\to \vee)_2}{\neg(p \vee \neg p), \neg(p \vee \neg p) \to}(\neg \to)}{\neg(p \vee \neg p) \to}(c \to)}{\to \neg\neg(p \vee \neg p)}(\to \neg)$$

However there are no ways to derive it without cut and contraction rules. The only possible derivation search tree is as follows:

$$\dfrac{\dfrac{\dfrac{\dfrac{\to p \quad \text{or} \quad \dfrac{p \to}{\to \neg p}(\to \neg)}{\to p \vee \neg p}(\to \vee)_1 \text{ or } (\to \vee)_2}{\neg(p \vee \neg p) \to}(\neg \to)}{\to \neg\neg(p \vee \neg p)}(\to \neg)$$

It is easy to derive sequent $\to p \vee \neg p$ in *GPR*, however it is not derivable in *GIN* (if $\to p$ and $\to \neg p$ are not derivable), because in the succedent of the premise of application of $(\to \vee)$ rule only one of the formulas p and $\neg p$ can occur (similar situation is demonstrated in Example 2.10.7). However, every sequent that is derivable in *GIN* is also derivable in *GPR*.

Let's analyse more examples.

Example 2.10.8. Derivations of sequents $\to \neg(p \wedge \neg p)$ and $\to \neg\neg\neg p \supset \neg p$ are as follows:

$$\dfrac{\dfrac{\dfrac{\dfrac{p \to p}{\neg p, p \to}(\neg \to)}{p, \neg p \to}(e \to)}{p \wedge \neg p \to}(\wedge \to)}{\to \neg(p \wedge \neg p)}(\to \neg) \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{p \to p}{\neg p, p \to}(\neg \to)}{p \to \neg\neg p}(\to \neg)}{\neg\neg\neg p, p \to}(\neg \to)}{p, \neg\neg\neg p \to}(e \to)}{\neg\neg\neg p \to \neg p}(\to \neg)}{\to \neg\neg\neg p \supset \neg p}(\to \supset)$$

Formula $\neg\neg p \supset p$ is not derivable in intuitionistic calculus (recall that axiom *4.3* is changed in *HIN*), however formula $p \supset \neg\neg p$ is derivable.

Example 2.10.9. Let's derive sequent $\to p \supset (\neg p \supset q)$ in *GIN*:

$$\dfrac{\dfrac{\dfrac{p \to p}{\neg p, p \to q}(\neg \to)}{p \to \neg p \supset q}(\to \supset)}{\to p \supset (\neg p \supset q)}(\to \supset)$$

Example 2.10.10. Let's derive sequent $\rightarrow (p \supset q) \supset (\neg q \supset \neg p)$ in *GIN*:

$$\dfrac{\dfrac{\dfrac{\dfrac{p \rightarrow p}{\neg q, p \rightarrow p}\,(w\rightarrow)}{p, \neg q \rightarrow p}\,(e\rightarrow) \qquad \dfrac{\dfrac{\dfrac{\dfrac{q \rightarrow q}{p, q \rightarrow q}\,(w\rightarrow)}{q, p \rightarrow q}\,(e\rightarrow)}{\neg q, q, p \rightarrow}\,(\neg\rightarrow)}{q, \neg q, p \rightarrow}\,(e\rightarrow)}{q, p, \neg q \rightarrow}\,(e\rightarrow)}{p \supset q, p, \neg q \rightarrow}\,(\supset\rightarrow)$$
$$\dfrac{p, p \supset q, \neg q \rightarrow}{p, \neg q, p \supset q \rightarrow}\,(e\rightarrow)$$
$$\dfrac{\neg q, p \supset q \rightarrow \neg p}{p \supset q \rightarrow \neg q \supset \neg p}\,(\rightarrow\supset)$$
$$\rightarrow (p \supset q) \supset (\neg q \supset \neg p)\,(\rightarrow\supset)$$

2.11 Relational algebra

Let's generalise the definition of a relation.

Definition 2.11.1. A *relation* between sets $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_n$ is any subset of Cartesian product $\mathcal{A}_1 \times \mathcal{A}_2 \times \cdots \times \mathcal{A}_n$. A relation is *finite* if the subset is finite.

The most straight forward way to define a relation is to enumerate all the elements of the subset. The same approach could be used to define the constants with which some predicate is true (see Example 2.1.4). Thus relations can also be presented as predicates. Another way to define a relation is to provide a table of the elements. This approach is the most common in relational algebra.

Although in the interpretation of predicate formula there is only one set $\mathcal{M}$ and the domain of all the variables of n-place predicate is the same, in an element of a relation each part usually describes a different property of the element and therefore is called an *attribute*. Each attribute of a relation can have a different domain.

Example 2.11.2. Let the relation Student describe the students of some university. Let's define it in set $\mathcal{N} \times \mathcal{E} \times \mathbb{N}$, where $\mathcal{N}$ is the set of names and $\mathcal{E}$ is the set of e-mails. A relation has three attributes: a name of the student, an e-mail and a year the student is in. The relation can be defined by enumerating all the elements: $\{$(Tom, tom@gmail.com, 1), (Helen, agirl@mail.lt, 2), (Peter, peter.griffin@hotmail.com, 2)$\}$ or by providing a table:

Student:

Name	*E-mail*	*Year of study*
Tom	tom@gmail.com	1
Helen	agirl@mail.lt	2
Peter	peter.griffin@hotmail.com	2

Predicate Student(x, y, z) is true if (x, y, z) is an element of the relation.

A relational algebra is presented as finite number of such finite relations and operators, which are used to produce new relations from the existing ones. The main operators of relational algebra are: projection, select, union, subtraction, Cartesian product, intersection, join:

1. *Projection.* Suppose P is a relation with attributes $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_n$. Then a projection $\pi_{\{\mathcal{A}_{i_1}, \mathcal{A}_{i_2}, \ldots, \mathcal{A}_{i_m}\}}(P)$, where $1 \leqslant i_1 < i_2 < \cdots < i_m \leqslant n$, is a relation with attributes $\mathcal{A}_{i_1}, \mathcal{A}_{i_2}, \ldots, \mathcal{A}_{i_m}$ and elements $(a_1, a_2, \ldots, a_m)$ such that $(x_1, x_2, \ldots, x_n) \in P$, where $x_{i_j} = a_j$ for every $j \in [1, m]$ and other x_k are some constants.

   In other words, a projection of P is a relation obtained by deleting columns that do not represent attributes from $\{\mathcal{A}_{i_1}, \mathcal{A}_{i_2}, \ldots, \mathcal{A}_{i_m}\}$. Duplicate rows are removed from the table.

P:

A	B	C
a	a	c
a	b	b
b	c	a
c	d	b

$\pi_{A,B}(P)$:

A	B
a	a
a	b
b	c
c	d

2. *Select* (denoted σ). Let P be a relation with attributes $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_n$. Then $\sigma_F(P)$ is a relation with attributes $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_n$ and elements of P that satisfy formula F. Here F is a formula that consists of terms connected by logical operators $\wedge$, $\vee$ or $\neg$. A term in this context is of the form $\mathcal{A}_i \theta \mathcal{A}_j$ or $\mathcal{A}_i \theta a$, where $\mathcal{A}_i$ and $\mathcal{A}_j, i, j \in [1, n]$ are some attributes, a is some constant and $\theta \in \{=, \neq, >, <, \leqslant, \geqslant\}$.

   In other words, $\sigma_F(P)$ is obtained from P by deleting the rows, which do not satisfy the formula.

   Let $F = \Big((A = a) \vee (A = c)\Big) \wedge \neg(B = b)$.

P:

A	B	C
a	a	c
a	b	b
b	c	a
c	d	b

$\sigma_F(P)$:

A	B	C
a	a	c
c	d	b

3. *Union.* Let P and Q be relations with attributes $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_n$. Then a union of P and Q (denoted $P \cup Q$) is a relation with attributes $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_n$ and elements $\{e : e \in P \text{ or } e \in Q\}$.

A union is obtained by supplementing a table of one relation with the rows of the table of the other one. Once again, duplicate rows are removed.

P:

A	B	C
a	a	c
a	b	b
b	c	a
c	d	b

Q:

A	B	C
d	d	d
a	a	c
a	b	b
d	c	a

$P \cup Q$:

A	B	C
a	a	c
a	b	b
b	c	a
c	d	b
d	d	d
d	c	a

4. *Subtraction* (denoted $-$). Let P and Q be relations with attributes $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_n$. Then $P-Q$ is a relation with attributes $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_n$ and elements $\{e : e \in P \text{ and } e \notin Q\}$.

   In other words $P - Q$ is obtained from P by removing rows that a present in Q.

P:

A	B	C
a	a	c
a	b	b
b	c	a
c	d	b

Q:

A	B	C
a	b	b
a	a	c
a	b	b
d	c	a

$P - Q$:

A	B	C
a	a	c
b	c	a
c	d	b

5. *Cartesian product.* Let P be a relation with attributes $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_n$ and Q be a relation with attributes $\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_m$. A Cartesian product $P \times Q$ is a relation with attributes $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_n, \mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_m$ and elements:

   $$\{(a_1, a_2, \ldots, a_n, b_1, b_2, \ldots, b_m) : (a_1, a_2, \ldots, a_n) \in P, (b_1, b_2, \ldots, b_m) \in Q\}$$

   In other words a table of Cartesian product composes of every possible combination of rows from tables of relation P and Q.

P:

A	B
a	a
a	b

Q:

C	D	E
d	d	d
a	a	c

$P \times Q$:

A	B	C	D	E
a	a	d	d	d
a	b	d	d	d
a	a	a	a	c
a	b	a	a	c

6. *Intersection.* Let P and Q be relations with attributes $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_n$. An intersection of P and Q (denoted $P \cap Q$) is a relation with attributes $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_n$ and elements $\{e : e \in P \text{ and } e \in Q\}$.

An intersection contains rows that are present in both tables.

P:

A	B	C
a	a	c
a	b	b
b	c	a
c	d	b

Q:

A	B	C
d	d	d
a	a	c
a	b	b
d	c	a

$P \cap Q$:

A	B	C
a	a	c
a	b	b

7. *Join.* Let P and Q be relations. Then a join is a select of Cartesian product of the two. In other words, a join $P \bowtie_F Q = \sigma_F(P \times Q)$. Assume $F = (B < D) \wedge \neg(A = 1)$.

P:

A	B	C
3	2	4
1	2	3
3	6	7

Q:

D	E
3	2
5	7

$P \bowtie_F Q$:

A	B	C	D	E
3	2	4	3	2
3	2	4	5	7

Relational algebra is used in relational databases. The described operations are used to define a query. However, queries are usually presented using SQL language. On the other hand, expressions of relational algebra or even formulas of predicate logic can also be used instead. Suppose P and Q are some relations with attributes A, B and C. An examples of a query could be $\exists x P(x, x, x)$ or $\exists z\Big(P(x, y, z) \wedge Q(z, y, x)\Big)$. If a query formula has no free variables (the former one), then the result of the query is either *true* or *false*. Otherwise, if query is defined using formula $F(x_1, x_2, \ldots, x_n)$ with free variables $x_1, x_2, \ldots, x_n$, then the result of the query is set of multiples $(a_1, a_2, \ldots, a_n)$ such that $F(a_1, a_2, \ldots, a_n)$ is true.

Example 2.11.3. Suppose P is a relation with attributes A, B and C defined as follows:

A	B	C
a	α	3
b	β	2
c	α	4
a	β	5

Let's define the same query using three different notations:

1. SQL: `SELECT A FROM P WHERE C>3`,
2. Expression of relational algebra: $\pi_A\Big(\sigma_{C>3}(P)\Big)$,
3. Formula of predicate logic: $\exists y \exists z\Big(P(x, y, z) \wedge (z > 3)\Big)$.

The result of a query presented as predicate formula in some cases can depend on the domains of the attributes of the relations. Let P be a relation with attributes A and B defined with a table:

A	B
a	a
a	b
b	a

Let's analyse the result of query $\neg P(x, y)$. It composes of all the pairs that are not part of the table. If the domain of both attributes is $\{a, b\}$, then the result of the query is composed of one element: $\{(b, b)\}$. If however a domain is $\{a, b, c\}$, then the result is $\{(a, c), (b, b), (b, c), (c, a), (c, b), (c, c)\}$. However, negation doesn't always result in such situation. E.g. a result of query $P(x, y) \wedge \neg P(x, x)$ doesn't depend on the domain and is always $\{(b, a)\}$.

There are many various requirements for a query, that ensure that it does not depend on the domain of the attributes of the relations. In 1972 E.F. Codd proved that if the query does not depend on the domain, then it is possible to present it as an expression of relational algebra.

Chapter 3

Deductive databases

3.1 Datalog

Deductive databases were created in a search of new ways to present data. Deductive databases consist of datasets together with rules which are used to obtain new data from the existing ones. The start of deductive databases is associated with C.C. Green and B. Raphael article *The use of theorem-proving techniques in question-answering systems*, presented in 1968.

In a conference in Toulouse (France) several papers were presented, which are fundamental in the field of deductive databases:

- R. Reiter article about the concept of a closed world;
- K.L. Clark article about the negation as a failure;
- J.M. Nicolas and K. Yazdanian article about checking he integrity conditions.

In this section one deductive database system, called *Datalog* (data + logic), is analysed. In databases of Datalog data sets consist of atomic predicate formulas and the rules are presented using predicate formulas of certain form. It is also assumed that at least some of the formulas contain some individual constant. Predicate formulas used in Datalog do not contain function variables. Thus a term in this context is individual variable or individual constant. Herbrand universe consists of all the constants of the deductive database.

Data and logic of deductive database can be presented as disjuncts of predicate logic. In this section let's analyse a special form of a disjunct, called *normal disjunct*: $(l_1 \wedge l_2 \wedge \ldots \wedge l_n) \supset F$. Here $l_i, i \in [1, n]$ are literals and F is some atomic formula. In Datalog syntax such formula is presented as expression $F :- l_1, l_2, \ldots, l_n$. An atomic formula F is called a *head* and formula $l_1, l_2, \ldots, l_n$ is called a *body*. If all the literals of the body are atomic formulas, then the formula is called *Horn disjunct*. If the body of the formula is empty (i.e. $n = 0$), then it is called a *fact*. In that case, symbol :− is omitted and simply F is written instead. Formulas without a head are also analysed. Such formulas are called *restrictions*. If, however, a formula contains both a head and a body, then it is called a *rule*.

Here only facts that do not contain individual variables are analysed. All the variables in the rules are assumed to be bounded by quantifier $\forall$. The quantifiers are omitted in Datalog syntax. Any set of facts, rules and restrictions is called a *deductive database*. If every rule of the database is Horn disjunct, then the database is called *Horn deductive database*.

Implication in the rule $(l_1 \wedge l_2 \wedge \ldots \wedge l_n) \supset F$ of a database has different meaning than in the propositional (or predicate) logic. It is said that F is *derivable* (*true*) only if all the $l_i, i \in [1, n]$ are derivable. More precisely, let $x_1, x_2, \ldots, x_m$ be all the individual variables of a rule. Then for some substitution $\alpha = \{{}^{y_1}/_{x_1}, {}^{y_2}/_{x_2}, \ldots, {}^{y_m}/_{x_m}\}$ formula $F\alpha$ is derivable, if every $l_i\alpha, i \in [1, n]$ is derivable. A fact is always derivable.

A query of such database is a conjunction of literals $l_1 \wedge l_2 \wedge \ldots \wedge l_n$, where $n \geqslant 0$. A query in Datalog is presented in a following way: $?{-}l_1 \wedge l_2 \wedge \ldots \wedge l_n$. If $n = 0$, then it is said that query is empty. Suppose $x_1, x_2, \ldots, x_m$ is a set of all the variables that occur in query $l_1 \wedge l_2 \wedge \ldots \wedge l_n$. Let's denote the query $P(x_1, x_2, \ldots, x_m)$. Then the result of the query is a set of multiples $(t_1, t_2, \ldots t_m)$ such that $P(t_1, t_2, \ldots t_m)$ is derivable from the database. If a set of variables of query is empty, then the result is simply *true* if the query is derivable and *false* otherwise.

Example 3.1.1. A deductive database can be defined as follows.

Facts:

$$\begin{array}{ll} \text{Father}(\text{John}, \text{Eve}) & \text{Mother}(\text{Christine}, \text{Eve}) \\ \text{Father}(\text{Peter}, \text{Monica}) & \text{Mother}(\text{Eve}, \text{Amy}) \end{array}$$

Rules:

$$\begin{array}{l} \text{Relative}(x, y) :- \text{Father}(x, y) \\ \text{Relative}(x, y) :- \text{Mother}(x, y) \\ \text{Relative}(x, z) :- \text{Mother}(x, y) \wedge \text{Relative}(y, z) \\ \text{Relative}(x, z) :- \text{Father}(x, y) \wedge \text{Relative}(y, z) \end{array}$$

Restrictions:

$$\begin{array}{l} :- \text{Father}(x, x) \\ :- \text{Mother}(x, x) \\ :- \text{Father}(x, y) \wedge \text{Mother}(x, z) \end{array}$$

Herbrand universe is $\{\text{John}, \text{Eve}, \text{Christine}, \text{Peter}, \text{Monica}, \text{Amy}\}$. Herbrand model is:

$$\begin{array}{ll} \text{Father}(\text{John}, \text{Eve}) & \text{Mother}(\text{Christine}, \text{Eve}) \\ \text{Father}(\text{Peter}, \text{Monica}) & \text{Mother}(\text{Eve}, \text{Amy}) \\ \text{Relative}(\text{John}, \text{Eve}) & \text{Relative}(\text{Christine}, \text{Eve}) \\ \text{Relative}(\text{Peter}, \text{Monica}) & \text{Relative}(\text{Eve}, \text{Amy}) \\ \text{Relative}(\text{Christine}, \text{Amy}) & \text{Relative}(\text{John}, \text{Amy}) \end{array}$$

A set of facts can also be presented as tables:

Father	
John	Eve
Peter	Monica

Mother	
Christine	Eve
Eve	Amy

There are three types of predicates in a database of Datalog:

1. *Extensional*. They are defined using only facts. The set of extensional predicates is called *extensional database* (denoted EDB).
2. *Intentional*. They are defined by rules. Every predicate, which occurs in a head of at least one rule is intentional. The set of intentional predicates is called *intentional database* (denoted IDB). IDB is also called a *Datalog program*.
3. Equality predicate = and other predicates already included in database syntax.

Using equality predicate in syntax of Datalog causes problems in derivation search algorithm. However it can be included into the IDB. Suppose, $P(x, y)$ is part of the Datalog program, then equality predicate (let's call it $E(x, y)$) with respect to predicate $P(x, y)$ can be defined using additional rules:

$$\begin{aligned} E(y, x) &:\!- E(x, y) \\ E(x, z) &:\!- E(x, y) \wedge E(y, z) \\ E(x, x) &:\!- P(x, y) \\ E(y, y) &:\!- P(x, y) \\ P(z, y) &:\!- P(x, y), E(x, z) \\ P(x, z) &:\!- P(x, y), E(y, z) \end{aligned}$$

To avoid usage of predicate =, which is built in the syntax of Datalog, equality predicate $E(x, y)$ must be defined with respect to every predicate of the program.

Let's first analyse *base Datalog* (it is also called *standard Datalog* or simply *Datalog*). Databases of Datalog satisfy these conditions:

- All the variables that are part of the head of some rule, are also part of the body of the rule;
- All the rules are Horn disjuncts, i.e. all the literals of the body of the rule are atomic formulas.

Suppose, $\mathcal{P}$ is a Datalog program and $\mathcal{I}$ is a set of some main intentional atomic formulas. Operator $\mathcal{T}_\mathcal{P}$ is defined as follows: $\mathcal{T}_\mathcal{P}(\mathcal{I})$ is a set of atomic formulas F such that there is a rule $F' \leftarrow G_1, G_2, \ldots, G_n$ in $\mathcal{P}$ and substitution α such that $F'\alpha = F$ and for every $i \in [1, n]$ formula $G_i\alpha \in \mathcal{I}$. If F is a fact, then $F \in \mathcal{T}_\mathcal{P}(\mathcal{I})$. If $\mathcal{T}_\mathcal{P}(\mathcal{I}) = \mathcal{I}$, then set $\mathcal{I}$ is called a *fixed point* of operator $\mathcal{T}_\mathcal{P}$.

Let $\mathcal{M}_\mathcal{P}$ be a set of all the main atomic formulas, that are derivable in program $\mathcal{P}$.

Some properties of operator $\mathcal{T}_\mathcal{P}$:

- it is monotonic, i.e. if $\mathcal{I}_1 \subseteq \mathcal{I}_2$, then $\mathcal{T}_\mathcal{P}(\mathcal{I}_1) \subseteq \mathcal{T}_\mathcal{P}(\mathcal{I}_2)$;
- there always is a smallest fixed point of $\mathcal{T}_\mathcal{P}$, which is a subset of every fixed point;

- $\mathcal{M}_{\mathcal{P}}$ is a smallest fixed point of $\mathcal{T}_{\mathcal{P}}$;

From the properties of $\mathcal{T}_{\mathcal{P}}$ it follows that the result of the query can be found by first finding $\mathcal{T}_{\mathcal{P}}(\varnothing)$, then $\mathcal{T}_{\mathcal{P}}\big(\mathcal{T}_{\mathcal{P}}(\varnothing)\big)$ etc…. Finally a set $\mathcal{M}_{\mathcal{P}}$ will be obtained. Such procedure is called naive or primitive search.

Example 3.1.2. Consider deductive database presented in Example 3.1.1. In that case:

$$\mathcal{T}_{\mathcal{P}}(\varnothing) = \{\text{Father}(\text{John}, \text{Eve}), \text{Mother}(\text{Christine}, \text{Eve}),$$

$$\text{Father}(\text{Peter}, \text{Monica}), \text{Mother}(\text{Eve}, \text{Amy})\}$$

$$\mathcal{T}_{\mathcal{P}}\big(\mathcal{T}_{\mathcal{P}}(\varnothing)\big) = \mathcal{T}_{\mathcal{P}}(\varnothing) \cup \{\text{Relative}(\text{John}, \text{Eve}), \text{Relative}(\text{Christine}, \text{Eve}),$$

$$\text{Relative}(\text{Peter}, \text{Monica}), \text{Relative}(\text{Eve}, \text{Amy})\}$$

$$\mathcal{T}_{\mathcal{P}}\Big(\mathcal{T}_{\mathcal{P}}\big(\mathcal{T}_{\mathcal{P}}(\varnothing)\big)\Big) = \mathcal{T}_{\mathcal{P}}\big(\mathcal{T}_{\mathcal{P}}(\varnothing)\big) \cup \{\text{Relative}(\text{Christine}, \text{Amy}),$$

$$\text{Relative}(\text{John}, \text{Amy})\}$$

Set $\mathcal{T}_{\mathcal{P}}\Big(\mathcal{T}_{\mathcal{P}}\big(\mathcal{T}_{\mathcal{P}}(\varnothing)\big)\Big)$ is a smallest fixed point of operator $\mathcal{T}_{\mathcal{P}}$ and also a Herbrand model.

A program is *recursive* if there is at least one rule, which contains at least one occurrence of an intentional predicate in its body. A program is called *linear*, if in the body of every rule there is no more than one occurrence of intentional predicate.

Example 3.1.3. Let the IDB be:

$$\begin{aligned} P(x, y) &:\!- Q(x), R(x, y), R(y, x) \\ P(x, x) &:\!- Q(x), S(x, x) \\ T(x) &:\!- R(x, x), P(x, y), S(y, x) \end{aligned}$$

Predicates P and T are intentional, predicates Q, R and S are extensional. Datalog program is recursive and linear.

Predicates of EDB are also called program *input predicates* and some of the predicates of IDB (usually not all of them are chosen) — *output predicates*.

A derivation of some main atomic formula F from deductive database according to program $\mathcal{P}$ can be presented as a tree. It is said that program $\mathcal{P}$ is *bounded*, if there is a natural number k such that the height of every derivation tree of any formula is no more than k. Number k is called (Datalog) program *upper bound*. It is said that boundedness problem of program $\mathcal{P}$ is *decidable* if it is possible to calculate the upper bound of the program. However in 1987 H. Gaifman, H. Mairson, Y. Sagiv and M.Y. Vardi proved that in general boundedness problem is undecidable.

3.2 Programs with negation operator

In standard Datalog if formula F is not derivable from database, then it is assumed that $\neg F$ is derivable.

Example 3.2.1. Suppose the database describes the flight schedule. The EDB is:

$$\begin{gathered}\text{Flight}(\text{Vilnius}, \text{Prague})\\ \text{Flight}(\text{Vilnius}, \text{Oslo})\\ \text{Flight}(\text{Vilnius}, \text{Tallinn})\\ \text{Flight}(\text{Vilnius}, \text{Helsinki})\end{gathered}$$

The IDB is empty and query is $?-\text{Flight}(\text{Vilnius}, \text{Beijing})$. Because predicate $\text{Flight}(\text{Vilnius}, \text{Beijing})$ is not derivable, it is assumed that formula $\neg\text{Flight}(\text{Vilnius}, \text{Beijing})$ is true.

Such interpretation of negation is called *closed world semantics*. It is assumed that $\neg F$ is true if it is not clear (not derivable, impossible to prove) that F is true.

If sets $\{l_1, \ldots, l_{i-1}, F, l_{i+1}, \ldots, l_n\}$ and $\{l_1, \ldots, l_{i-1}, \neg F, l_{i+1}, \ldots, l_n\}$ are Herbrand models of one database, then it is said that they are *independent* from atomic formula F and instead of two models, only one is used: $\{l_1, \ldots, l_{i-1}, l_{i+1}, \ldots, l_n\}$. Let's analyse only Herbrand models, in which negative atomic formulas are omitted and there are no atomic formulas from which they are independent. Suppose that $\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_m$ are Herbrand models. Model $\mathcal{I}_j$ is called *minimal* if it is a subset of every model, i.e. $\mathcal{I}_j \subseteq \mathcal{I}_k$ for every $k \in [1, m]$.

If rules that are not Horn disjuncts are analysed, then it is possible to have several minimal models for one database.

Example 3.2.2. Suppose database is:

$$\begin{gathered}P(a)\\ Q(x) :- P(x), \neg R(x)\\ R(x) :- P(x), \neg Q(x)\end{gathered}$$

This database has two models: $\{P(a), Q(a)\}$ and $\{P(a), R(a)\}$. Therefore, it is impossible to find a single minimal model.

To avoid such situations, usually only the databases in which intentional predicates are divided into different levels are analysed. M.N. Emden and R.A. Kowalski proved that if standard Datalog is analysed (i.e. the IDB consists of Horn disjuncts only), then it always has a single minimal model.

Let's expand the database concept to allow rules with literals (not only atomic formulas) as their heads. In such case semantically all the atomic formulas can be divided into three sets: true (atomic formulas are derivable), false (negations of atomic formulas are derivable) and undefined (neither formulas nor their negations are derivable) formulas. Such concept is called *open world semantics*: it is assumed that everything that is not derivable is unknown.

Definition 3.2.3. It is said that a rule of Datalog deductive database is *safe*, if every variable in the head of the rule is part of at least one positive atomic formula in the body of the rule.

Example 3.2.4. The rule $P(a, x) \leftarrow Q(b, a), R(a, c)$ is not safe, because x is not part of the body of the rule. It is not clear with which constants instead of x atomic formula $P(a, x)$ is derivable. E.g. can the value of x be chosen outside of Herbrand universe?

To avoid such confusion usually only databases with safe rules are analysed.

3.3 Datalog and relational algebra

The data of relational algebra usually defined in SQL can also be defined using Datalog.

Example 3.3.1. Predicate master_students(name, surname) definition in SQL:

```
 CREATE VIEW master_students AS
SELECT name,surname FROM student
WHERE type_of_studies = 'master'
```

can be defined using a rule of Datalog:

$$\text{master\_students(name,surname)} :-$$

$$\text{student(name, surname, type)}, \text{type} = \text{master}$$

Let's define how seven operators of relational algebra can be defined in Datalog. They are discussed in more detail in Section 2.11.

- Projection $R(x_{i_1}, x_{i_2}, \ldots, x_{i_m})$ of relation $P(x_1, x_2, \ldots, x_n)$, such that $x_{i_j} \in \{x_1, x_2, \ldots, x_n\}, j \in [1, m]$ and $1 \leqslant i_1 < i_2 < \cdots < i_m \leqslant n$:

$$R(x_{i_1}, x_{i_2}, \ldots, x_{i_m}) :- P(x_1, x_2, \ldots, x_n)$$

- Select operation can be demonstrated using an example. Say that relation $P(x, y, z)$ is defined and values of x and z must be selected such that $y = a$. Such select can be expressed using expression of relational algebra $\sigma_{y=a}(P)$ or in Datalog:

$$R(x, z) :- P(x, a, z)$$

- Union of relations $P(x_1, x_2, \ldots, x_n)$ and $Q(x_1, x_2, \ldots, x_n)$:

$$R(x_1, x_2, \ldots, x_n) :- P(x_1, x_2, \ldots, x_n)$$
$$R(x_1, x_2, \ldots, x_n) :- Q(x_1, x_2, \ldots, x_n)$$

- Difference of relations $P(x_1, x_2, \ldots, x_n)$ and $Q(x_1, x_2, \ldots, x_n)$:

$$R(x_1, x_2, \ldots, x_n) :- P(x_1, x_2, \ldots, x_n), \neg Q(x_1, x_2, \ldots, x_n)$$

- Cartesian product of relations $P(x_1, x_2, \ldots, x_n)$ and $Q(y_1, y_2, \ldots, y_m)$:

$$R(x_1, x_2, \ldots, x_n, y_1, y_2, \ldots, y_m) :- P(x_1, x_2, \ldots, x_n), Q(y_1, y_2, \ldots, y_m)$$

- Intersection of relations $P(x_1, x_2, \ldots, x_n)$ and $Q(x_1, x_2, \ldots, x_n)$:

$$R(x_1, x_2, \ldots, x_n) :- P(x_1, x_2, \ldots, x_n), Q(x_1, x_2, \ldots, x_n)$$

- Only one type of join is demonstrated:

$$R(x_1, x_2, \ldots, x_n, y_1, y_2, \ldots, y_m, z_1, z_2, \ldots, z_k) :-$$

$$P(x_1, x_2, \ldots, x_n, y_1, y_2, \ldots, y_m), Q(y_1, y_2, \ldots, y_m, z_1, z_2, \ldots, z_k)$$

3.4 Disjunctive Datalog

Disjunctive Datalog expands the definitions even further. The disjuncts in disjunctive Datalog are of the form

$$F_1 \vee F_1 \vee \ldots \vee F_n :- G_1, G_2, \ldots, G_m, \neg H_1, \neg H_2, \ldots, \neg H_k$$

where $F_i, i \in [1, n]$, $G_j, j \in [1, m]$ and $H_l, l \in [1, k]$ are atomic formulas. As well as in standard Datalog disjuncts without a body are called *facts*. If a disjunct contains both a body and a head, then it is called a *rule*. The set of all the facts is called *extensional database* (or EDB) and the set of all the rules — *intentional database* (or IDB).

A *query* of database of disjunctive Datalog is a conjunction of literals: $?- \; l_1 \wedge l_2 \wedge \ldots \wedge l_s$. An *interpretation* of database of disjunctive Datalog is some subset of atomic formulas of Herbrand base of the database. It is said that atomic formula without variables F is *true* with interpretation $\mathcal{I}$ (denoted $\mathcal{I} \vDash F$) if $F \in \mathcal{I}$. Otherwise, it is said that F is *false* with interpretation $\mathcal{I}$ (denoted $\mathcal{I} \nvDash F$). Literal $\neg F$ is true with interpretation $\mathcal{I}$ iff $\mathcal{I} \nvDash F$.

Definition 3.4.1. Suppose $\mathcal{D}$ is some deductive database of disjunctive Datalog ($\mathcal{D} = \text{EDB} \cup \text{IDB}$). Then set Ground($\mathcal{D}$) consists of formulas $F\alpha$ such that $F \in \mathcal{D}$ and $\alpha = \{{}^{a_1}/_{x_1}, {}^{a_2}/_{x_2}, \ldots, {}^{a_n}/_{x_n}\}$, where $a_i \in \mathcal{H}, i \in [1, n]$, set $\mathcal{H}$ is Herbrand universe of $\mathcal{D}$ and $x_1, x_2, \ldots, x_n$ is a full list of individual variables in F.

For better explanation let's analyse the example.

Example 3.4.2. Let the database $\mathcal{D}$ be:

$$R(a)$$
$$S(x) \vee P(x,b) :- R(x)$$
$$S(x) :- \neg R(x), \neg Q(a,x)$$
$$S(x) :- P(x,b)$$

In this case

$$\text{Ground}(\mathcal{D}) = \{R(a), S(a) \vee P(a,b) :- R(a), S(b) \vee P(b,b) :- R(b),$$
$$S(a) :- \neg R(a), \neg Q(a,a), S(b) :- \neg R(b), \neg Q(a,b),$$
$$S(a) :- P(a,b), S(b) :- P(b,b)\}$$

It is said that interpretation $\mathcal{I}$ of deductive database $\mathcal{D}$ *satisfies* formula $F \in \text{Ground}(\mathcal{D})$ if the following holds:

- if F is a fact, then $\mathcal{I} \vDash F$;
- if F is a rule of the form $G_1 \vee G_1 \vee \ldots \vee G_n :- l_1, l_2, \ldots, l_m$ and $\forall i \in [1,m] : \mathcal{I} \vDash l_i$, then there is $j \in [1,n]$ such that $\mathcal{I} \vDash G_j$.

Interpretation is called a *model* of database $\mathcal{D}$ if it satisfies every formula of $\text{Ground}(\mathcal{D})$. A model is *minimal* if it is not a subset of any other model.

Example 3.4.3. Database presented in Example 3.4.2 contains many models, e.g. $\{R(a), S(a), P(b,a), Q(a,a), Q(a,b)\}$ or $\{R(a), S(a), S(b), P(a,a)\}$. However only two of them are minimal: $\mathcal{I}_1 = \{R(a), S(a), Q(a,b)\}$ and $\mathcal{I}_2 = \{R(a), S(a), S(b)\}$.

In some cases only rules with positive atomic formulas (i.e. of the form $F_1 \vee F_1 \vee \ldots \vee F_n :- G_1, G_2, \ldots, G_m$, where $F_i, i \in [1,n]$ and $G_j, j \in [1,m]$ are atomic formulas) are analysed. A database, which contains only such rules is called a *positive database*. An *answer set* of positive database is any minimal model of the database.

Example 3.4.4. Let the database be:

Facts	Rules	
$p \vee q$	$c :- q$	$d :- b, c$
$q \vee r$	$d :- a$	$d \vee l :- b$
	$b :- q$	$a :- p, r$

In this case propositional variables can stand for some atomic formulas without individual variables. The models of such deductive database are e.g. $\mathcal{M}_1 = \{p, r, a, d\}$, $\mathcal{M}_2 = \{q, b, c, d\}$ and $\mathcal{M}_3 = \{q, b, c, d, l\}$. However only $\mathcal{M}_1$ and $\mathcal{M}_2$ are minimal and $\mathcal{M}_3$ is not. Thus $\mathcal{M}_1$ and $\mathcal{M}_2$ are answer sets.

However to define an answer set in any database of disjunctive Datalog another definition is needed.

Definition 3.4.5. Let $\mathcal{D}$ be a database of disjunctive Datalog and $\mathcal{I}$ be an interpretation of $\mathcal{D}$. Then database $\mathcal{D}^{\mathcal{I}}$ is obtained from Ground($\mathcal{D}$) by removing all the formulas with the body that contains literal $\neg F$ such that $F \in \mathcal{I}$ and by removing all the negative literals (formulas of the form $\neg G$) from other formulas.

It should be noted, that $\mathcal{D}^{\mathcal{I}}$ is always a positive database.

Definition 3.4.6. Let $\mathcal{D}$ be a database of disjunctive Datalog then a minimal model $\mathcal{I}$ is an answer set of $\mathcal{D}$ if it is an *answer set* of $\mathcal{D}^{\mathcal{I}}$.

Example 3.4.7. Let's continue with database considered in Examples 3.4.2 and 3.4.2. In that case

$$\mathcal{D}^{\mathcal{I}_1}\{R(a), S(a) \vee P(a,b) :- R(a), S(b) \vee P(b,b) :- R(b),$$
$$S(a) :- P(a,b), S(b) :- P(b,b)\}$$
$$\mathcal{D}^{\mathcal{I}_2} = \{R(a), S(a) \vee P(a,b) :- R(a), S(b) \vee P(b,b) :- R(b),$$
$$S(b), S(a) :- P(a,b), S(b) :- P(b,b)\}$$

Interpretation $\mathcal{I}_2$ is a minimal model of $\mathcal{D}^{\mathcal{I}_2}$, however $\mathcal{D}^{\mathcal{I}_1}$ has even smaller model than $\mathcal{I}_1$ — $\{R(a), S(a)\}$. Thus only $\mathcal{I}_2$ is an answer set of $\mathcal{D}$.

Suppose that $l_1 \wedge l_2 \wedge \ldots \wedge l_s$ is a query for database of disjunctive Datalog without individual variables. It is said that interpretation $\mathcal{I}$ *satisfies* the query if $\mathcal{I} \vDash l_i$ for every $i \in [1, s]$. There are two possible approaches to deciding what is the result of the query. One is called *brave reasoning* and according to it the result of a query without individual variables is positive if at least one answer set of the database satisfies the query. According to *cautious reasoning* the result is positive if all the answer sets satisfy the query. Finally if $x_1, x_2, \ldots, x_n$ is a full list of individual variables of query $l_1 \wedge l_2 \wedge \ldots \wedge l_s$, then the result is a set of formulas $(l_1 \wedge l_2 \wedge \ldots \wedge l_s)\alpha$, where $\alpha = \{{}^{a_1}/_{x_1}, {}^{a_2}/_{x_2}, \ldots, {}^{a_n}/_{x_n}\}$ is a substitution, $a_j \in \mathcal{H}, j \in [1, n]$ and $\mathcal{H}$ is Herbrand universe, such that the result of query $(l_1 \wedge l_2 \wedge \ldots \wedge l_s)\alpha$ is positive.

Definition 3.4.8. It is said that database is stratified, if the set of all the predicates of the database can be divided into subsets $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_n$ such that $\mathcal{A}_i \cap \mathcal{A}_j = \varnothing, \forall i \neq j$ and if some predicate $P \in \mathcal{A}_i$ is part of the head of some rule of the database, then every other predicate, which is part of the head of the same rule is part of $\mathcal{A}_i$. Moreover, if some predicate $Q \in \mathcal{A}_j$ is part of the body of that rule, then $j \leqslant i$ if the occurrence of the predicate is positive and $j < i$ if the occurrence is negative. The set $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_n$ is called a stratification of the database.

Example 3.4.9. The following database is stratified:

$$P(a,c) \vee R(c)$$
$$S(x) \leftarrow \neg R(x), \neg P(a,x)$$
$$S(x) \vee T(x,c) \leftarrow R(x)$$
$$S(x) \leftarrow T(x,b)$$
$$V(x) \vee Q(y) \leftarrow \neg T(x,y), P(x,a)$$

The stratification is $\mathcal{A}_1 = \{P, R\}$, $\mathcal{A}_2 = \{S, T\}$ and $\mathcal{A}_3 = \{V, Q\}$.

3.5 U-Datalog

U-Datalog additionally defines a concept of *update operation*. Databases of U-Datalog can be changed during the execution of the query. There are two types of update operations:

- *insert*, denoted $+F$;
- *delete*, denoted $-F$.

Here F is an atomic formula.

A database of U-Datalog together with extensional and intentional databases contains a set of *active rules*. An active rule is an expression of the form $U_1, U_2, \ldots, U_n, l_1, l_2, \ldots, l_m \to V_1, V_2, \ldots, V_k$, where $U_i, i \in [1, n]$ and $V_j, j \in [1, k]$ are update operations and $l_t, t \in [1, k]$ are literals.

A concept of query is replaced by *transaction* — a sequence of update operations without individual variables. A transaction is executed following these steps until the database doesn't change:

1. A set of update operations $\mathcal{U}$ consists of every operation of the transaction.
2. EDB is updated. For every update operation $U \in \mathcal{U}$ if $U = +F$, then formula F is added into EDB, if $U = -F$, then formula F is removed from EDB.
3. Active rules are used to create a new set of update operations. For every active rule $U_1, U_2, \ldots, U_n, l_1, l_2, \ldots, l_m \to V_1, V_2, \ldots, V_k$ if for some substitution α every $U_i\alpha \in \mathcal{U}, i \in [1, n]$ and every $l_j\alpha, j \in [1, m]$ is derivable from EDB and IDB, then $V_t\alpha \in \mathcal{U}'$ for every $t \in [1, k]$. Set $\mathcal{U}'$ is filled with every update operation generated by every active rule.
4. If $\mathcal{U}' \neq \varnothing$, then set $\mathcal{U}'$ is a new set of update operations $\mathcal{U}$ and the algorithm is repeated from step 2.

Example 3.5.1. The extensional database is:

stud(John)	exam(Algebra, John)
stud(Peter)	exam(Logics, Peter)
failed(Logics, John)	nogrant(John)
failed(Algebra, Peter)	nogrant(Peter)

The intentional database is:

$$\text{firstyear}(x) :- \text{exam}(\text{Algebra}, x), \text{exam}(\text{Logics}, x)$$

The active rules are:

$$+\text{exam}(x, y), \text{stud}(y) \to -\text{failed}(x, y)$$
$$\text{firstyear}(x) \to -\text{nogrant}(x)$$

John and Peter are students. To complete the first year of their studies they had to pass algebra and logics exams. John passed algebra but failed in logics, therefore he haven't completed the first year of his studies and will not get a grant for the following year. Peter, however, contrary passed logics and failed algebra exams. Nevertheless he also haven't completed the first year and will not receive a grant.

Suppose that transaction is +exam(Algebra, Peter), i.e. Peter passed algebra exam later. The new extensional database is:

stud(John)	exam(Algebra, John)
stud(Peter)	exam(Logics, Peter)
failed(Logics, John)	nogrant(John)
exam(Algebra, Peter)	

Chapter 4

Modal logics

4.1 Syntax and semantics

Many various propositions can defined in predicate logic, however it is not straight forward how to define concepts such as knowledge, belief or time. Consider the following examples:

- Proposition *the population of Lithuania is between 3 and 4 millions* is true for know, but it is possible that it will be false later. How to define formula that this proposition is *always* true?
- Proposition *the largest planet of the Solar system is Jupiter* is true, but not everybody knows that. How to define formula that everybody *knows* that this proposition is true?
- It is not known if proposition *the afterlife exists* is true or not. It is a matter of belief. How to define formula that someone *believes* that this proposition is true?

To formalise these propositions two new operators (called *modal operators*) are introduced: $\Box$, defining necessity, and $\Diamond$, defining possibility.

Definition 4.1.1. A *modal formula* is defined in a following way:

- propositional variable is a formula;
- if F is a formula, then $\neg F$, $\Box F$ and $\Diamond F$ are also formulas.
- if F and G are formulas, then $(F \vee G)$, $(F \wedge G)$, $(F \supset G)$ and $(F \leftrightarrow G)$ are also formulas.

The priority of unary operators (negation $\neg$ and two modal operators $\Box$ and $\Diamond$) is the highest. Priorities of other operators are the same as in propositional logic case. Similarly to propositional logic, some parentheses can be omitted.

Example 4.1.2. These expressions are modal formulas: $\Box\big(p \supset (\Box q \wedge \Diamond\Diamond p)\big)$, $\Box\Diamond p$, $\neg p \vee \Box\Box(q \vee p)$.

Semantically modal operators can have many different meanings and the meaning depends on the application. Some of the possibilities are:

$\Box F$	$\Diamond F$
Someone (or some agent) knows that F is true	Someone (or some agent) does not know that F is false
Someone is sure that F is true	Someone thinks that F can possibly be true
Always F	Sometimes F
F will certainly occur	Event F is possible
F is provable	F is not improvable
All the paths of non-deterministic calculus with input data F end in final states	There is a path of non-deterministic calculus with input data F that ends in final state

Instead of interpretation of propositional logic, to interpret modal formulas Kripke structure is used.

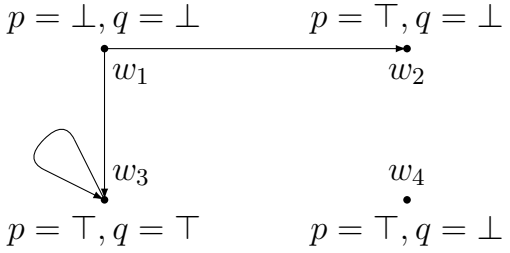
Definition 4.1.3. A *Kripke structure* of modal formula F is a triple $\mathcal{S} = \langle \mathcal{W}, \mathcal{R}, \nu \rangle$, where:

- $\mathcal{W} \neq \varnothing$ is some set, called the set of possible worlds.
- $\mathcal{R}$ is a binary relation between elements of set $\mathcal{W}$.
- ν is an interpretation function, which depends on the elements of $\mathcal{W}$. I.e. $\nu : \mathcal{P} \times \mathcal{W} \to \{\top, \bot\}$, where $\mathcal{P}$ is a set of propositional variables of formula F.

A pair $\langle \mathcal{W}, \mathcal{R} \rangle$ of Kripke structure $\langle \mathcal{W}, \mathcal{R}, \nu \rangle$ is called a *frame*.

Kripke structure can be displayed using oriented graph. Worlds of the structure are the vertices of the graph and the relation is presented using the oriented edges of the graph. To display an interpretation, value of every propositional variable can be displayed next to the name of the world in the graph. Another approach is to display only the names of those variables, which are true in that world.

Example 4.1.4. $\mathcal{W} = \{w_1, w_2, w_3, w_4\}$, $\mathcal{R} = \{(w_1, w_2), (w_1, w_3), (w_3, w_3)\}$. Suppose F has only two variables p and q and ν is defined as follows: in world $\nu(p, w_1) = \nu(q, w_1) = \bot$, $\nu(p, w_2) = \top$, $\nu(q, w_2) = \bot$, $\nu(p, w_3) = \nu(q, w_3) = \top$, $\nu(p, w_4) = \top$ and $\nu(q, w_4) = \bot$. Then structure (let's denote it $\mathcal{S}$) can be expressed using graph:



Similarly to propositional logic the fact that modal formula F is true with interpretation $\mathcal{S}$ in world w is denoted $\mathcal{S}, w \vDash F$ and is defined recursively.

Definition 4.1.5. Suppose $\mathcal{S} = \langle \mathcal{W}, \mathcal{R}, \nu \rangle$ is a Kripke structure of formula F and $w \in \mathcal{W}$, then:

- if F is a propositional variable, then $\mathcal{S}, w \vDash F$ iff $\nu(F, w) = \top$.
- if $F = \neg G$, then $\mathcal{S}, w \vDash F$ iff $\mathcal{S}, w \nvDash G$.
- if $F = G \wedge H$, then $\mathcal{S}, w \vDash F$ iff $\mathcal{S}, w \vDash G$ and $\mathcal{S}, w \vDash H$.
- if $F = G \vee H$, then $\mathcal{S}, w \nvDash F$ iff $\mathcal{S}, w \nvDash G$ and $\mathcal{S}, w \nvDash H$.
- if $F = G \supset H$, then $\mathcal{S}, w \nvDash F$ iff $\mathcal{S}, w \vDash G$ and $\mathcal{S}, w \nvDash H$.
- if $F = G \leftrightarrow H$, then $\mathcal{S}, w \vDash F$ iff $\mathcal{S}, w \vDash G$ and $\mathcal{S}, w \vDash H$ or $\mathcal{S}, w \nvDash G$ and $\mathcal{S}, w \nvDash H$.
- if $F = \Box G$, then $\mathcal{S}, w \vDash F$ iff for every world $w_1 \in \mathcal{W}$ such that $w \mathcal{R} w_1$ it is true that $\mathcal{S}, w_1 \vDash G$.
- if $F = \Diamond G$, then $\mathcal{S}, w \vDash F$ iff there is a world $w_1 \in \mathcal{W}$ such that $w \mathcal{R} w_1$ and $\mathcal{S}, w_1 \vDash G$.

Example 4.1.6. Let's analyse structure $\mathcal{S}$ defined in Example 4.1.4. Formula $\Diamond q$ is true in w_1 of that structure (i.e. $\mathcal{S}, w_1 \vDash \Diamond q$). The following expressions are also true: $\mathcal{S}, w_1 \vDash \Diamond \neg q$; $\mathcal{S}, w_1 \nvDash \Box q$; $\mathcal{S}, w_1 \vDash \Box p$; $\mathcal{S}, w_1 \nvDash \Box p \supset p$; $\mathcal{S}, w_3 \vDash \Box \Box p$; $\mathcal{S}, w_4 \nvDash \Box p \supset \Diamond p$.

If formula F is true in world w of structure $\mathcal{S}$ (i.e. $\mathcal{S}, w \vDash p$), then a pair $\langle \mathcal{S}, w \rangle$ is called a *model* of formula F. If F is true in every world of structure $\mathcal{S}$, then it is said that F is *valid* in structure $\mathcal{S}$ (denoted $\mathcal{S} \vDash F$). If there is a world w such that $\mathcal{S}, w \vDash F$, then it is said that F is *satisfiable* in structure $\mathcal{S}$.

Let F be some formula, $\mathcal{S}$ be a Kripke structure of F and w a world of $\mathcal{S}$. Then a *local model checking* aims to find if $\mathcal{S}, w \vDash F$ (i.e. if $\langle \mathcal{S}, w \rangle$ is a model of F). A *global model checking* aims to find if formula F is valid in structure $\mathcal{S}$. And the aim of *mixed model checking* is to find every world w' of structure $\mathcal{S}$ such that $\mathcal{S}, w' \vDash F$.

Example 4.1.7. Suppose $\mathcal{S} = \langle \mathcal{W}, \mathcal{R}, \nu \rangle$ is a Kripke structure. A set of worlds $\mathcal{W} = \{w_0, w_1, w_2, \dots\}$. A relation between worlds is: $w_n \mathcal{R} w_{n+2}$, $w_n \mathcal{R} w_{n+3}$ and $w_n \mathcal{R} w_{n+5}$ for all $n \in [0, \infty)$. Finally an interpretation function ν is defined as follows: $\nu(p, w_i) = \top$, iff i is even number, $\nu(q, w_j) = \top$ iff j is odd number and $\nu(r, w_k) = \top$, iff k is prime number. Let's check if the following formulas are satisfiable in structure $\mathcal{S}$:

1. $\Box r$;
2. $\Box \Diamond q$;
3. $\Box\big(p \vee (q \vee \Box r)\big)$.

The solutions are:

1. The aim is to find a world in which $\Box r$ is true. Because of the definition of $\mathcal{R}$, the world w_i must be found such that $\mathcal{S}, w_{i+2} \vDash r$, $\mathcal{S}, w_{i+3} \vDash r$ and $\mathcal{S}, w_{i+5} \vDash r$. Now taking into account definition of $\nu(r, w)$, the number $i \in [0, \infty]$ must be fount such that $i + 2$, $i + 3$ and $i + 5$ are prime numbers. The only such number is 0, because 2, 3 and 5 are primes. Therefore $\Box r$ is satisfiable in structure $\mathcal{S}$, because $\mathcal{S}, w_0 \vDash \Box r$.

2. Once again, some number i must be found such that for every number $j \in \{i+2, i+3, i+5\}$ the following would be true: at least one of $j+2$, $j+3$ or $j+5$ is odd number. It is easy to see that for any j either $j+2$ or $j+3$ is odd (and another one is even). This fact does not depend neither on the value of j nor on the value of i. Thus formula $\Box\Diamond q$ is not only satisfiable, it is valid in structure $\mathcal{S}$.

3. Every number is either odd or even. Thus formula $p \vee q$ is true in every world of $\mathcal{W}$. Moreover, for the same reason $\Box(p \vee q)$ is true in every world of $\mathcal{W}$. It is not hard to argue that if $p \vee q$ is true in some world, then $p \vee q \vee \Box r$ is also true in that world. And thus, $\Box(p \vee q \vee \Box r)$ is true in every world of $\mathcal{M}$. Now additional parentheses do not make any difference, because $p \vee (q \vee \Box r) \equiv (p \vee q) \vee \Box r$. Therefore $\Box\big(p \vee (q \vee \Box r)\big)$ is valid in structure $\mathcal{S}$ and thus also satisfiable.

Example 4.1.8. Suppose $\mathcal{S} = \langle \mathcal{W}, \mathcal{R}, \nu \rangle$ is a Kripke structure. Let $\mathcal{W}$ be a set of all the lines of the plain. Two lines $x, y \in \mathcal{W}$ are in a relation $x\mathcal{R}y$ iff x and y are perpendicular or parallel and $x \neq y$. Finally interpretation ν is defined as follows: $\nu(p, w) = \top$ iff line w crosses axis O_x. Let's check if the following formulas are satisfiable in structure $\mathcal{S}$:

1. $\Box\Diamond p$;

2. $\Box\Box p \vee \Diamond\Diamond p$.

It is easy to show that both formulas are valid in structure $\mathcal{S}$. From the validity, satisfiability follows immediately.

1. Let w be some line of the plain, $p(w)$ be a set of lines that are perpendicular to w and $q(w)$ be a set of lines that are parallel to w, $w \notin q(w)$. Then the aim is to prove that $\Diamond p$ is true in every world of $p(w) \cup q(w)$. Let's take $w' \in p(w) \cup q(w)$. Formula $\Diamond p$ is true in world w' if at least one line from $p(w') \cup q(w')$ crosses O_x axis. If w is neither parallel nor perpendicular to O_x axis, then neither are lines from $p(w) \cup q(w)$. Therefore for any $w' \in p(w) \cup q(w)$ all the lines of $p(w') \cup q(w')$ crosses O_x axis. If w is parallel to O_x axis, then for any $w' \in p(w)$ all the lines of $q(w')$ crosses O_x axis. Moreover, for any $w' \in q(w)$ all the lines in $p(w')$ crosses O_x axis. Finally, if w is perpendicular to O_x, then for every $w' \in p(w)$, any line from $p(w')$ crosses O_x axis, and for every $w' \in q(w)$, any line from $q(w')$ crosses O_x axis. Therefore, formula $\Box\Diamond p$ is true in any world $w \in \mathcal{W}$ and thus it is valid in structure $\mathcal{S}$.

2. It is enough to show that $\Diamond\Diamond p$ is valid in structure $\mathcal{S}$. From that fact, the validity of $\Box\Box p \vee \Diamond\Diamond p$ follows immediately. Let w be some line of the plain. Let's take any line $w' \in p(w) \cup q(w)$. The aim is to show, that there is a line in $p(w') \cup q(w')$, that crosses O_x axis. This can be proved by following the same discussion as in the previous case.

   It can be noted, that formula $\Box\Box p$ is indeed satisfiable in $\mathcal{S}$, however it is not valid. Formula is true in world w (with line w), if for any line w', which is perpendicular or parallel to w ($w' \neq w$), it is true that any line, which is perpendicular or parallel to w' (except w'), crosses O_x axis. This is true, if w is neither parallel, nor perpendicular to O_x. If however w is parallel to O_x, then let's take some $w' \in p(w)$. In this case no line from $p(w')\backslash\{O_x\}$ crosses O_x axis and thus $\mathcal{S}, w \not\models \Box\Box p$.

Now let's analyse several other modal formulas.

Theorem 4.1.9. *The following formulas are valid in any Kripke structure.*

1. $\Diamond F \supset \neg\Box\neg F$;
2. $\neg\Box\neg F \supset \Diamond F$;
3. $\Box F \supset \neg\Diamond\neg F$;
4. $\neg\Diamond\neg F \supset \Box F$;
5. $\Box(F \supset G) \supset (\Box F \supset \Box G)$.

Here F and G are any modal formulas.

Proof. To prove the validity of the formulas suppose that Kripke structure $\mathcal{S} = \langle \mathcal{W}, \mathcal{R}, \nu \rangle$ is analysed and $w \in \mathcal{W}$ is some world of the structure. Let's show that these formulas are true in world w of such structure (i.e. $\langle \mathcal{S}, w \rangle$ is a model of these formulas).

1. If $\mathcal{S}, w \not\models \Diamond F$, then because of the definition of implication, $\langle \mathcal{S}, w \rangle$ is a model of $\Diamond F \supset \neg\Box\neg F$. Suppose that $\mathcal{S}, w \models \Diamond F$. Then there is $w' \in \mathcal{W}$ such that $w\mathcal{R}w'$ and $\mathcal{S}, w' \models F$. Therefore, $\mathcal{S}, w' \not\models \neg F$. Now because of that $\mathcal{S}, w \not\models \Box\neg F$ and thus $\mathcal{S}, w \models \neg\Box\neg F$.

2. If $\mathcal{S}, w \not\models \neg\Box\neg F$, then $\langle \mathcal{S}, w \rangle$ is a model of $\neg\Box\neg F \supset \Diamond F$. Suppose $\mathcal{S}, w \models \neg\Box\neg F$. Then $\mathcal{S}, w \not\models \Box\neg F$. Now there is $w' \in \mathcal{W}$ such that $w\mathcal{R}w'$ and $\mathcal{S}, w' \not\models \neg F$. Therefore, $\mathcal{S}, w' \models F$. Finally because of that $\mathcal{S}, w \models \Diamond F$.

3. If $\mathcal{S}, w \not\models \Box F$, then $\langle \mathcal{S}, w \rangle$ is a model of $\Box F \supset \neg\Diamond\neg F$. Suppose that $\mathcal{S}, w \models \Box F$. Then for all $w' \in \mathcal{M}$ such that $w\mathcal{R}w'$ it is true that $\mathcal{S}, w' \models F$. Therefore, $\mathcal{S}, w' \not\models \neg F$. Now because of that $\mathcal{S}, w \not\models \Diamond\neg F$ and thus $\mathcal{S}, w \models \neg\Diamond\neg F$.

4. If $\mathcal{S}, w \not\models \neg\Diamond\neg F$, then $\langle \mathcal{S}, w \rangle$ is a model of $\neg\Diamond\neg F \supset \Box F$. Suppose $\mathcal{S}, w \models \neg\Diamond\neg F$. Then $\mathcal{S}, w \not\models \Diamond\neg F$. Now there are no $w' \in \mathcal{W}$ such that $w\mathcal{R}w'$ and $\mathcal{S}, w' \models \neg F$, so for all $w' \in \mathcal{W}$ such that $w\mathcal{R}w'$ it is true that $\mathcal{S}, w' \not\models \neg F$. Therefore, $\mathcal{S}, w' \models F$. Finally because of that $\mathcal{S}, w \models \Box F$.

5. If $\mathcal{S}, w \nvDash \Box(F \supset G)$, then $\langle \mathcal{S}, w \rangle$ is a model of $\Box(F \supset G) \supset (\Box F \supset \Box G)$. Suppose $\mathcal{S}, w \vDash \Box(F \supset G)$. Therefore for any $w' \in \mathcal{W}$ such that $w\mathcal{R}w'$ it is true that $\mathcal{S}, w' \vDash F \supset G$. Because of that, if for all such w' it is true that $\mathcal{S}, w' \vDash F$, then it is also true that $\mathcal{S}, w' \vDash G$. From this it follows that $\mathcal{S}, w \vDash \Box F$ and $\mathcal{S}, w \vDash \Box G$, thus $\mathcal{S}, w \vDash \Box F \supset \Box G$. If however there is a world w' such that $\mathcal{S}, w' \nvDash F$, then $\mathcal{S}, w \nvDash \Box F$ and therefore $\mathcal{S}, w \vDash \Box F \supset \Box G$.

□

Corollary 4.1.10. *It is possible to use only one modal operator. Usually* $\Box$ *is chosen and formulas of the form* $\Diamond F$ *are replaced by* $\neg\Box\neg F$.

Definition 4.1.11. A *projection* of modal formula F into propositional logic $\mathrm{Proj}(F)$ is obtained from F by deleting all the occurrences of modal operators.

Example 4.1.12. If $F = p \wedge (\Box\Diamond q \vee \neg\Box p)$, then $\mathrm{Proj}(F) = p \wedge (q \vee \neg p)$.

If formula $\mathrm{Proj}(F)$ is not valid in propositional logic, then there is a Kripke structure in which modal formula F is not valid too. To prove that let ν be a propositional interpretation with which $\nu \nvDash \mathrm{Proj}(F)$. Now let's construct Kripke structure $\mathcal{S} = \langle \mathcal{W}, \mathcal{R}, \nu' \rangle$, where $\mathcal{W} = \{w\}$, $\mathcal{R} = \{(w, w)\}$ and $\nu'(p, w) = \nu(p)$ for any propositional variable p of F. It is not hard to check inductively that $\mathcal{S}, w \nvDash F$.

However, the opposite statement (*if* $\mathrm{Proj}(F)$ *is valid in propositional logic, then* F *is valid in every Kripke structure*) is not always true. E.g. propositional formula $(p \vee q) \supset (p \vee q)$ is valid in propositional logic, however $\Box(p \vee q) \supset (\Box p \vee \Box q)$ is not valid in Kripke structure $\mathcal{S} = \langle \mathcal{W}, \mathcal{R}, \nu \rangle$, where $\mathcal{W}$ is a set of natural numbers, $x\mathcal{R}y$ iff $y = x + 1$ or $y = x + 2$, $\nu(p, w) = \top$ iff w is an even number and $\nu(q, w) = \top$ iff w is an odd number. It is easy to check that $\mathcal{S} \nvDash \Box(p \vee q) \supset (\Box p \vee \Box q)$.

Let's analyse properties of the relations that are defined in Definition 1.4.2. It is obvious that all of them can be defined in predicate logic. Moreover, some of the properties can be defined using modal formulas. Suppose that $\mathcal{S} = \langle \mathcal{M}, \mathcal{R}, \mathcal{V} \rangle$ is a Kripke structure, then:

- relation $\mathcal{R}$ is reflexive, iff formula $\Box F \supset F$ is valid in structure $\mathcal{S}$ for any formula F;
- relation $\mathcal{R}$ is complete, iff formula $\Box F \supset \Diamond F$ is valid in structure $\mathcal{S}$ for any formula F;
- relation $\mathcal{R}$ is symmetric, iff formula $F \supset \Box\Diamond F$ is valid in structure $\mathcal{S}$ for any formula F;
- relation $\mathcal{R}$ is transitive, iff formula $\Box F \supset \Box\Box F$ is valid in structure $\mathcal{S}$ for any formula F;
- relation $\mathcal{R}$ is euclidean, iff formula $\Box F \supset \Box\Diamond F$ is valid in structure $\mathcal{S}$ for any formula F;

These formulas have their names. They add interesting properties of modal operators, because of that they are included as axioms in some modal logics. The axioms are summarised in the following table:

Name	Formula	$\mathcal{R}$ is:
(T)	$\Box F \supset F$	reflexive
(D)	$\Box F \supset \Diamond F$	complete
(B)	$F \supset \Box\Diamond F$	symmetric
(4)	$\Box F \supset \Box\Box F$	transitive
(5)	$\Box F \supset \Box\Diamond F$	euclidean

The logics are defined in the next chapter.

4.2 Hilbert-type calculi

As it was mentioned earlier, only one of two modal operators can be analysed. Thus the Hilbert-type calculus is provided for formulas containing $\Box$ modal operator only. The calculus is obtained by adding modal axioms and a modal rule to propositional Hilbert-type calculus:

Definition 4.2.1. Hilbert-type calculus for modal logics consists of the same axioms and rules as *HPC*, Necessity Generalisation rule (*NG*):

$$\frac{F}{\Box F}$$

And axioms that depend on modal logic in question:

(K): $\Box(F \supset G) \supset (\Box F \supset \Box G)$;

(T): $\Box F \supset F$;

(D): $\Box F \supset \Diamond F$;

(B): $F \supset \Box\Diamond F$;

(4): $\Box F \supset \Box\Box F$;

(5): $\Box F \supset \Box\Diamond F$.

Different modal logics are obtained by choosing a different set of modal axioms. Propositional axioms, *MP* and *NG* rules are included in all the calculi.

The different modal logics are summarised in the following table:

Logics	Hilbert-type calculus	Modal axioms
K	*HK*	*(K)*
T	*HT*	*(K)*, *(T)*
S4	*HS4*	*(K)*, *(T)*, *(4)*
S5	*HS5*	*(K)*, *(T)*, *(4)*, *(5)*
K4	*HK4*	*(K)*, *(4)*
B	*HB*	*(K)*, *(B)*
D	*HD*	*(K)*, *(D)*
K45	*HK45*	*(K)*, *(4)*, *(5)*
KD45	*HKD45*	*(K)*, *(D)*, *(4)*, *(5)*

The table is interpreted as follows: Hilbert-type calculus for modal logic *S4* (denoted *HS4*) contains axioms of *HPC*, rules *MP* and *NG* and modal axioms *(K)*, *(T)* and *(4)*. Modal logics that include axiom *(K)* are called *normal*, the other ones are called *semi normal*. It is known that if modal logic consists of axiom *(K)* and any combination of axioms *(T)*, *(D)*, *(4)*, *(5)* and *(B)*, then it is decidable.

Example 4.2.2. Let's find a derivation of formula $p \supset \Diamond p$ in calculus *HT*. First of all, $\Diamond$ is not part of the calculus, therefore it must be replaced and a derivation of formula $p \supset \neg\Box\neg p$ must be found:

1. $\Box\neg p \supset \neg p$	Axiom *(T)*, $\{^{\neg p}/_{F}\}$.
2. $(\Box\neg p \supset \neg p) \supset (\neg\neg p \supset \neg\Box\neg p)$	Axiom *4.1*, $\{^{\Box\neg p}/_{F}, ^{\neg p}/_{G}\}$.
3. $\neg\neg p \supset \neg\Box\neg p$	*MP* rule from 1 and 2.
4. $p \supset \neg\neg p$	Axiom *4.2*, $\{^{p}/_{F}\}$.
5. $(\neg\neg p \supset \neg\Box\neg p) \supset \big(p \supset (\neg\neg p \supset \neg\Box\neg p)\big)$	Axiom *1.1*, $\{^{\neg\neg p \supset \neg\Box\neg p}/_{F}, ^{p}/_{G}\}$.
6. $p \supset (\neg\neg p \supset \neg\Box\neg p)$	*MP* rule from 3 and 5.
7. $\big(p \supset (\neg\neg p \supset \neg\Box\neg p)\big) \supset \big((p \supset \neg\neg p) \supset (p \supset \neg\Box\neg p)\big)$	Axiom *1.2*, $\{^{p}/_{F}, ^{\neg\neg p}/_{G}, ^{\neg\Box\neg p}/_{H}\}$.
8. $(p \supset \neg\neg p) \supset (p \supset \neg\Box\neg p)$	*MP* rule from 6 and 7.
9. $p \supset \neg\Box\neg p$	*MP* rule from 4 and 8.

Now let's analyse the following example.

Example 4.2.3. Let's say that $\mathcal{S} = \langle \mathcal{W}, \mathcal{R}, \nu \rangle$ is a Kripke structure for formula $F = \Box p \supset p$, where $\mathcal{W} = \{w_1, w_2\}$, $\mathcal{R} = \{(w_1, w_2)\}$, $\nu(p, w_1) = \bot$ and $\nu(p, w_2) = \top$. It is possible to demonstrate the structure graphically:

Now let's check, if $\mathcal{S}, w_1 \vDash F$. According to the definition $\mathcal{S}, w_1 \vDash \Box p$, because $\mathcal{S}, w_2 \vDash p$ and w_2 is the only world such that $w_1 \mathcal{R} w_2$. However $\mathcal{S}, w_1 \nvDash p$. Therefore $\mathcal{S}, w_1 \nvDash \Box p \supset p$.

This example demonstrates, that it is possible to construct a Kripke structure, in which axiom *(T)* is not valid. Indeed, as stated in previous section, for this axiom to be valid in each world of some Kripke structure $\mathcal{S} = \langle \mathcal{W}, \mathcal{R}, \nu \rangle$, the relation $\mathcal{R}$ of $\mathcal{S}$ must be reflexive. Therefore, the validity of formula in modal logic (the fact that formula F is valid in modal logic $\mathcal{L}$ is denoted $\vDash_{\mathcal{L}} F$) is defined taking into account the requirements of the axioms, that are part of the logic:

- $\vDash_{K} F$ iff for any Kripke structure $\mathcal{S}$ it is true that $\mathcal{S} \vDash F$.
- $\vDash_{T} F$ iff for any Kripke structure $\mathcal{S}$ with reflexive relation $\mathcal{S} \vDash F$.
- $\vDash_{K4} F$ iff for any Kripke structure $\mathcal{S}$ with transitive relation $\mathcal{S} \vDash F$.
- $\vDash_{S4} F$ iff for any Kripke structure $\mathcal{S}$ with reflexive and transitive relation $\mathcal{S} \vDash F$.

Definitions of $\vDash_{S5} F$, $\vDash_{B} F$, $\vDash_{D} F$, $\vDash_{K45} F$ and $\vDash_{KD45} F$ are obtained analogously by taking into account the requirements of modal axioms of logics.

Analogously, the satisfiability of formulas in modal logics is defined:

- F is satisfiable in modal logic K iff it is satisfiable in some Kripke structure $\mathcal{S}$.
- F is satisfiable in modal logic T iff it is satisfiable in some Kripke structure $\mathcal{S}$ with reflexive relation.
- F is satisfiable in modal logic *K4* iff it is satisfiable in some Kripke structure $\mathcal{S}$ with transitive relation.
- F is satisfiable in modal logic *S4* iff it is satisfiable in some Kripke structure $\mathcal{S}$ with reflexive and transitive relation.

This definition can be extended to other logics too.

4.3 Sequent calculi

Definition 4.3.1. Gentzen-type calculus for modal logics is obtained from *GPC* by adding modal rules, which depend on logic:

Calculus *GK* for modal logic *K*:

$$\frac{\Gamma_2 \to F}{\Gamma_1, \Box\Gamma_2 \to \Delta, \Box F}\ (\to \Box)$$

Calculus *GK4* for modal logic *K4*:

$$\frac{\Gamma_2, \Box\Gamma_2 \to F}{\Gamma_1, \Box\Gamma_2 \to \Delta, \Box F}\ (\to \Box)$$

Calculus *GT* for modal logic *T*:

$$\frac{F, \Box F, \Gamma \to \Delta}{\Box F, \Gamma \to \Delta}\ (\Box \to) \qquad \frac{\Gamma_2 \to F}{\Gamma_1, \Box\Gamma_2 \to \Delta, \Box F}\ (\to \Box)$$

Calculus *GS4* for modal logic *S4*:

$$\frac{F, \Box F, \Gamma \to \Delta}{\Box F, \Gamma \to \Delta}\ (\Box \to) \qquad \frac{\Box\Gamma_2 \to F}{\Gamma_1, \Box\Gamma_2 \to \Delta, \Box F}\ (\to \Box)$$

Here $\Box\Gamma_2$ is a set of formulas, starting with $\Box$. In that case Γ_2 is a set of formulas obtained from $\Box\Gamma_2$ by removing outermost $\Box$ occurrence in each formula.

Example 4.3.2. Let's derive sequent $\rightarrow \neg\Box\neg(p\vee\Box\neg p)$ in sequent calculus *GS4*.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{p,\Box\neg(p\vee\Box\neg p)\rightarrow p,\Box\neg p}{p,\Box\neg(p\vee\Box\neg p)\rightarrow p\vee\Box\neg p}(\rightarrow\vee)}{p,\neg(p\vee\Box\neg p),\Box\neg(p\vee\Box\neg p)\rightarrow}(\neg\rightarrow)}{p,\Box\neg(p\vee\Box\neg p)\rightarrow}(\Box\rightarrow)}{\Box\neg(p\vee\Box\neg p)\rightarrow\neg p}(\rightarrow\neg)}{\Box\neg(p\vee\Box\neg p)\rightarrow p,\Box\neg p}(\rightarrow\Box)}{\Box\neg(p\vee\Box\neg p)\rightarrow p\vee\Box\neg p}(\rightarrow\vee)}{\neg(p\vee\Box\neg p),\Box\neg(p\vee\Box\neg p)\rightarrow}(\neg\rightarrow)}{\Box\neg(p\vee\Box\neg p)\rightarrow}(\Box\rightarrow)}{\rightarrow\neg\Box\neg(p\vee\Box\neg p)}(\rightarrow\neg)$$

The main formula of the application of $(\Box\rightarrow)$ rule is repeated in the premiss too. This repetition cannot be omitted, because without it the sequent wouldn't be derivable:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{p\rightarrow}{\rightarrow\neg p}(\rightarrow\neg)}{\rightarrow p,\Box\neg p}(\rightarrow\Box)}{\rightarrow p\vee\Box\neg p}(\rightarrow\vee)}{\neg(p\vee\Box\neg p)\rightarrow}(\neg\rightarrow)}{\Box\neg(p\vee\Box\neg p)\rightarrow}}{\rightarrow\neg\Box\neg(p\vee\Box\neg p)}(\rightarrow\neg)$$

Example 4.3.3. Let's derive sequent $\Box(p\wedge q)\rightarrow\Box p\wedge\Box q$ in sequent calculus of logic *T*.

$$\dfrac{\dfrac{\dfrac{p,q\rightarrow p}{p\wedge q\rightarrow p}(\wedge\rightarrow)}{\Box(p\wedge q)\rightarrow\Box p}(\rightarrow\Box)\qquad\dfrac{\dfrac{p,q\rightarrow q}{p\wedge q\rightarrow q}(\wedge\rightarrow)}{\Box(p\wedge q)\rightarrow\Box q}(\rightarrow\Box)}{\Box(p\wedge q)\rightarrow\Box p\wedge\Box q}(\rightarrow\wedge)$$

Sometimes it is not enough to analyse only modal operator $\Box$. One such case is when it is required that negation be only in front of propositional variables. Every modal formula can be transformed into such form, however, operator $\Diamond$ is necessary. It is easy to extend a calculus to deal with modal operator $\Diamond$ (it must be kept in mind that $\Diamond F$ can be changed by $\neg\Box\neg F$). E.g. modal rules of sequent calculus for logic *S4* are as follows:

rules for operator $\Box$:

$$\dfrac{F,\Box F,\Gamma\rightarrow\Delta}{\Box F,\Gamma\rightarrow\Delta}(\Box\rightarrow)\qquad\dfrac{\Box\Gamma_2\rightarrow\Diamond\Delta_2,F}{\Gamma_1,\Box\Gamma_2\rightarrow\Delta_1,\Diamond\Delta_2,\Box F}(\rightarrow\Box)$$

rules for operator $\Diamond$:

$$\dfrac{\Box\Gamma_2,F\rightarrow\Diamond\Delta_2}{\Gamma_1,\Box\Gamma_2,\Diamond F\rightarrow\Delta_1,\Diamond\Delta_2}(\Diamond\rightarrow)\qquad\dfrac{\Gamma\rightarrow\Delta,F,\Diamond F}{\Gamma\rightarrow\Delta,\Diamond F}(\rightarrow\Diamond)$$

Here $\Diamond\Delta_2$ is a set of formulas, starting with $\Diamond$.

4.4 Tableaux calculus

Tableaux calculus for modal logics is similar to the one for predicate logic defined in Section 2.6. However instead of regular formulas, *prefixed formulas* are used. A prefixed formula is an expression of the form $\sigma\ F$, where σ is a finite sequence of natural numbers, called a *prefix*, and F is some formula. To derive formula F using tableaux calculus, the derivation search must start with prefixed formula $1\ \neg F$.

Definition 4.4.1. A tableaux calculus for modal logics consists of the following rules:

Conjunction rules:

$$\frac{\sigma\ F \wedge G}{\begin{array}{c}\sigma\ F\\ \sigma\ G\end{array}} \qquad \frac{\sigma\ \neg(F \vee G)}{\begin{array}{c}\sigma\ \neg F\\ \sigma\ \neg G\end{array}} \qquad \frac{\sigma\ \neg(F \supset G)}{\begin{array}{c}\sigma\ F\\ \sigma\ \neg G\end{array}}$$

Disjunction rules:

$$\frac{\sigma\ \neg(F \wedge G)}{\sigma\ \neg F \mid \sigma\ \neg G} \qquad \frac{\sigma\ F \vee G}{\sigma\ F \mid \sigma\ G} \qquad \frac{\sigma\ F \supset G}{\sigma\ \neg F \mid \sigma\ G}$$

Double negation rule:

$$\frac{\sigma\ \neg\neg F}{\sigma\ F}$$

Necessity rules:

$$\frac{\sigma\ \Box F}{\sigma.n\ F} \qquad \frac{\sigma\ \neg\Diamond F}{\sigma.n\ \neg F}$$

where $\sigma.n$ is any prefix, $n \in \mathbb{N}$.

Possibility rules:

$$\frac{\sigma\ \Diamond F}{\sigma.k\ F} \qquad \frac{\sigma\ \neg\Box F}{\sigma.k\ \neg F}$$

where $\sigma.k$ is a new prefix, that does not appear in the branch, $k \in \mathbb{N}$.

And additional rules for different modal logics:

Rules *(T)*:

$$\frac{\sigma\ \Box F}{\sigma\ F} \qquad \frac{\sigma\ \neg\Diamond F}{\sigma\ \neg F}$$

Rules *(D)*:

$$\frac{\sigma\ \Box F}{\sigma\ \Diamond F} \qquad \frac{\sigma\ \neg\Diamond F}{\sigma\ \neg\Box F}$$

Rules *(B)*:

$$\frac{\sigma.n\ \Box F}{\sigma\ F} \qquad \frac{\sigma.n\ \neg\Diamond F}{\sigma\ \neg F}$$

where $\sigma.n$ is any prefix, $n \in \mathbb{N}$.

Rules *(4)*:

$$\frac{\sigma\ \Box F}{\sigma.n\ \Box F} \qquad \frac{\sigma\ \neg\Diamond F}{\sigma.n\ \neg\Diamond F}$$

where $\sigma.n$ is any prefix, $n \in \mathbb{N}$.

Rules *(4r)*:

$$\frac{\sigma.n\ \Box F}{\sigma\ \Box F} \qquad\qquad \frac{\sigma.n\ \neg\Diamond F}{\sigma\ \neg\Diamond F}$$

where $\sigma.n$ is any prefix, $n \in \mathbb{N}$.

Once again, additional rules used in calculus for each logic are summarised in a table:

Logics	Hilbert-type calculus	Additional rules
K	*TK*	
T	*TT*	*(T)*
S4	*TS4*	*(T)*, *(4)*
S5	*TS5*	*(T)*, *(4)*, *(4r)*
K4	*TK4*	*(4)*
B	*TB*	*(B)*, *(T)*, *(4)*
D	*TD*	*(D)*

A branch is *closed*, if it contains two formulas $\sigma\ F$ and $\sigma\ \neg F$. A closed tree with $1\ \neg F$ as a root is called a derivation of modal formula F in tableaux calculus.

Example 4.4.2. Let's derive formula $F = (\Box\Diamond p \wedge \Box\Diamond q) \supset \Box\Diamond(\Box\Diamond p \wedge \Box\Diamond q)$ in tableaux calculus *TS4*:

$$1\ \neg\Big((\Box\Diamond p \wedge \Box\Diamond q) \supset \Box\Diamond(\Box\Diamond p \wedge \Box\Diamond q)\Big) = F_1$$
$$1\ \Box\Diamond p \wedge \Box\Diamond q = F_2$$
$$1\ \neg\Box\Diamond(\Box\Diamond p \wedge \Box\Diamond q) = F_3$$
$$1\ \Box\Diamond p = F_4$$
$$1\ \Box\Diamond q = F_5$$
$$1.1\ \neg\Diamond(\Box\Diamond p \wedge \Box\Diamond q) = F_6$$
$$1.1\ \neg(\Box\Diamond p \wedge \Box\Diamond q) = F_7$$

$$1.1\ \neg\Box\Diamond p = F_8 \quad 1.1\ \neg\Box\Diamond q = F_9$$
$$1.1\ \Box\Diamond p = F_{10} \quad 1.1\ \Box\Diamond q = F_{11}$$

Formulas F_2 and F_3 are obtained by applying conjunction rule to formula F_1. In the same way formulas F_4 and F_5 are obtained from F_2. To obtain formula F_6, possibility rule is applied to F_3. Formula F_7 results from the application of rule *(T)* to F_6. Formulas F_8 and F_9 are the result of application of disjunction rule to formula F_7. Finally, F_{10} is obtained from F_4 and F_{11} is obtained from F_5 after application of *(4)* rule.

One branch of derivation tree includes prefixed formulas $1.1\ \neg\Box\Diamond p$ and $1.1\ \Box\Diamond p$, the other one — $1.1\ \neg\Box\Diamond q$ and $1.1\ \Box\Diamond q$. Thus the tree is closed and it is a derivation of formula F.

If only formulas containing $\neg$, $\wedge$, $\vee$, $\Box$ and $\Diamond$ operators are analysed and negation is always in front of propositional variables, then tableaux system can be simplified. Here such systems for logics *S4* and *S5* are presented.

Definition 4.4.3. Simplified tableaux calculus for logic *S4* (denoted $TS4_s$) contains only one variant of each conjunction, disjunction, necessity, possibility, *(T)* and *(4)* rule:

$$\frac{\sigma\ F \wedge G}{\begin{array}{c}\sigma\ F\\ \sigma\ G\end{array}} \quad \frac{\sigma\ F \vee G}{\sigma\ F \mid \sigma\ G} \quad \frac{\sigma\ \Box F}{\sigma.n\ F} \quad \frac{\sigma\ \Diamond F}{\sigma.k\ F} \quad \frac{\sigma\ \Box F}{\sigma\ F} \quad \frac{\sigma\ \Box F}{\sigma.n\ \Box F}$$

As in *TS4* case, $\sigma.n$ is any index, $\sigma.k$ is a new index, that doesn't appear in the branch, $n, k \in \mathbb{N}$.

Definition 4.4.4. Simplified tableaux calculus for logic *S5* (denoted $TS5_s$) can be obtained by enriching calculus $TS4_s$ with one variant of *(4r)* rule:

$$\frac{\sigma.n\ \Box F}{\sigma\ \Box F}$$

Simplified tableaux calculi for other modal logics can be obtained from original tableaux calculi in analogous way.

4.5 Relation to predicate logic

For any modal formula F it is possible to find a predicate formula $\mathrm{Tr}(F)_x$ with one free variable x such that F is satisfiable in logic K iff $\mathrm{Tr}(F)_x$ is satisfiable in predicate logic. Let's demonstrate how formula $\mathrm{Tr}(F)_x$ can be obtained:

- $\mathrm{Tr}(G)_x = P(x)$, if G is a propositional variable. Here P is a predicate variable into which propositional variable G is transformed. The predicate variable is different for different propositional variables of modal formula, however it is the same for different occurrences of the same propositional variable.
- $\mathrm{Tr}(\neg G)_x = \neg\, \mathrm{Tr}(G)_x$;
- $\mathrm{Tr}(G \wedge H)_x = \mathrm{Tr}(G)_x \wedge \mathrm{Tr}(H)_x$;
- $\mathrm{Tr}(G \vee H)_x = \mathrm{Tr}(G)_x \vee \mathrm{Tr}(H)_x$;
- $\mathrm{Tr}(G \supset H)_x = \mathrm{Tr}(G)_\tau \supset \mathrm{Tr}(H)_x$;
- $\mathrm{Tr}(\Box G)_x = \forall y\Big(R(x,y) \supset \mathrm{Tr}(G)_y\Big)$, here y is a new variable and R is a predicate variable, which represents the relation of Kripke structure.
- $\mathrm{Tr}(\Diamond G)_x = \exists y\Big(R(x,y) \wedge \mathrm{Tr}(G)_y\Big)$, here y is a new variable.

Example 4.5.1. Let's find $\mathrm{Tr}\Big(\Box\Diamond(p \supset q)\Big)_x$:

$$\mathrm{Tr}\Big(\Box\Diamond(p \supset q)\Big)_x = \forall y\Big(R(x,y) \supset \mathrm{Tr}\Big(\Diamond(p \supset q)\Big)_y\Big) =$$

$$= \forall y\Big(R(x,y) \supset \exists z\Big(R(y,z) \wedge \mathrm{Tr}(p \supset q)_z\Big)\Big) =$$

$$= \forall y\Big(R(x,y) \supset \exists z\Big(R(y,z) \wedge \big(\operatorname{Tr}(p)_z \supset \operatorname{Tr}(q)_z\big)\Big)\Big) =$$

$$= \forall y\Big(R(x,y) \supset \exists z\Big(R(y,z) \wedge \big(P(z) \supset Q(z)\big)\Big)\Big)$$

This means that every modal formula can be written and analysed in predicate logic. However there are predicate formulas which cannot be defined using modal logic. Thus predicate logic is more expressive than modal logic. Nevertheless modal logic is expressive enough for some applications and it is much easier to solve many problems in modal logic than in predicate logic. For comparison a complexity of two problems is presented:

	Model checking	Satisfiability checking
Predicate logic	PSPACE	Impossible to solve
Modal logic	P	PSPACE

To define similar transformations for formulas of other modal logics, the requirements of the axioms must be taken into consideration. Let $\sigma \in \{D, T, K4, S4, B, S5, K45, KD45\}$, then F_σ is a conjunction of predicate formulas, that describe the properties of axioms of modal logic σ. The formulas are presented in the following table:

Property	Formula
Reflexivity	$\forall x R(x,x)$
Completeness	$\forall x \exists y R(x,y)$
Symmetricity	$\forall x \forall y\big(R(x,y) \supset R(y,x)\big)$
Transitivity	$\forall x \forall y \forall z\Big(\big(R(x,y) \wedge R(y,z)\big) \supset R(x,z)\Big)$
Euclidicity	$\forall x \forall y \forall z\Big(\big(R(x,y) \wedge R(x,z)\big) \supset R(y,z)\Big)$

E.g. if $\sigma = S4$, which contains axioms *(T)* and *(4)*, then formulas for reflexivity and transitivity must be included into F_σ, thus

$$F_\sigma = \forall x R(x,x) \wedge \forall x \forall y \forall z\Big(\big(R(x,y) \wedge R(y,z)\big) \supset R(x,z)\Big)$$

Modal formula G is satisfiable in logic σ iff predicate formula $\exists x \operatorname{Tr}(G)_x \wedge F_\sigma$ is satisfiable in predicate logic.

Modal formula G is valid if $\neg G$ is not satisfiable. Thus G is valid iff $\exists x \operatorname{Tr}(\neg G)_x \wedge F_\sigma$ is not satisfiable. Therefore, G is valid iff $\forall x \operatorname{Tr}(G)_x \vee \neg F_\sigma$ is valid.

It was mentioned that complexity of satisfiability checking in modal logic is PSPACE. In fact, when the relation has additional properties, complexity can be even lower:

Logics	Complexity
K, *D*, *T*, *K4*, *S4*	PSPACE
KD45, *S5*	NP

4.6 Mints transformation

Definition 4.6.1. Formulas F and G are *equivalent* (denoted $F \equiv G$) in modal logic $\mathcal{L}$, if for any Kripke structure $\mathcal{S}$ that satisfy the requirements of $\mathcal{L}$ and any world w of $\mathcal{S}$ $\mathcal{S}, w \vDash F$ iff $\mathcal{S}, w \vDash G$.

In modal logic *S4* these equivalences hold:

- $\Diamond\Diamond p \equiv \Diamond p$;
- $\Box\Box p \equiv \Box p$;
- $\Box\Diamond\Box\Diamond p \equiv \Box\Diamond p$;
- $\Diamond\Box\Diamond\Box p \equiv \Diamond\Box p$;
- $\neg\Diamond p \equiv \Box\neg p$;
- $\Box(p \wedge q) \equiv \Box p \wedge \Box q$;
- $\Diamond(p \vee q) \equiv \Diamond p \vee \Diamond q$;

However, $\Box(p \vee q) \not\equiv \Box p \vee \Box q$ and $\Diamond(p \wedge q) \not\equiv \Diamond p \wedge \Diamond q$.

In propositional logic the following statement is correct: if $G \equiv H$, then $F(G) \equiv F(H)$ for any formula F. This is not true in modal logics. E.g. from the fact that $p \equiv q$ doesn't follow that $\Box p \equiv \Box q$, because it is impossible to derive the following sequent:

$$\dfrac{\dfrac{\cdots \qquad \dfrac{\cdots \qquad \dfrac{\dfrac{\dfrac{\dfrac{\Box p \to q}{p, \Box p, q, p \to \Box q}\,(\to \Box)}{\Box p, q, p \to \Box q}\,(\Box \to)}{q, p \to \Box p \supset \Box q}\,(\to \supset) \qquad \cdots}{q, p \to (\Box p \supset \Box q) \wedge (\Box q \supset \Box p)}\,(\to \wedge)}{q, q \supset p \to (\Box p \supset \Box q) \wedge (\Box q \supset \Box p)}\,(\supset \to)}{p \supset q, q \supset p \to (\Box p \supset \Box q) \wedge (\Box q \supset \Box p)}\,(\supset \to)}{(p \supset q) \wedge (q \supset p) \to (\Box p \supset \Box q) \wedge (\Box q \supset \Box p)}\,(\wedge \to)$$

Although the given derivation search tree is in calculus *GS4*, similar trees could be constructed for other calculi.

Theorem 4.6.2 (Mints)**.** *For any formula F and its subformula G, in modal logic GS4 from the fact that $\Box(G \equiv H)$ follows that $F(G) \equiv F(H)$.*

Definition 4.6.3. *Modal literals* are literals of propositional logic and formulas of the form $\Box l$ and $\Diamond l$, where l is a literal of propositional logic.

Definition 4.6.4. A *modal disjunct* is a disjunction of modal literals.

Theorem 4.6.5. *For any formula F there is a set of modal disjuncts $G_1, G_2, \ldots, G_n$ and a literal of propositional logic l such that $\to F$ is derivable in sequent calculus GS4, iff $\Box G_1, \Box G_2, \ldots, \Box G_n, l \to$ is derivable.*

Let's define a transformation algorithm. First of all, let's take sequent $\to F$. Let's replace subformulas of the form $\neg p$, $p \vee q$, $p \wedge q$, $p \supset q$, $\Box p$, $\Diamond p$ by new variables one by one. The change requires that formula $\Box(r \leftrightarrow \neg p)$, $\Box\big(r \leftrightarrow (p \wedge q)\big)$, $\Box\big(r \leftrightarrow (p \vee q)\big)$, $\Box\big(r \leftrightarrow (p \supset q)\big)$, $\Box(r \leftrightarrow \Box p)$ or $\Box(r \leftrightarrow \Diamond p)$ respectively be included into antecedent (here r is a variable, which replaces the subformula). All of these formulas can be transformed into form $\Box G$, where G is a modal disjunct:

- $\Box(r \leftrightarrow \neg p) \equiv \Box\big((r \supset \neg p) \wedge (\neg p \supset r)\big) \equiv \Box\big((\neg r \vee \neg p) \wedge (p \vee r)\big) \equiv \Box(\neg r \vee \neg p) \wedge \Box(p \vee r)$ — formulas $\Box(\neg r \vee \neg p)$ and $\Box(p \vee r)$ are obtained;
- $\Box\big(r \leftrightarrow (p \wedge q)\big) \equiv \Box\Big(\big(r \supset (p \wedge q)\big) \wedge \big((p \wedge q) \supset r\big)\Big) \equiv \Box\Big(\big(\neg r \vee (p \wedge q)\big) \wedge \big(\neg(p \wedge q) \vee r\big)\Big) \equiv \Box\big((\neg r \vee p) \wedge (\neg r \vee q) \wedge (\neg p \vee \neg q \vee r)\big) \equiv \Box(\neg r \vee p) \wedge \Box(\neg r \vee q) \wedge \Box(\neg p \vee \neg q \vee r)$ — formulas $\Box(\neg r \vee p)$, $\Box(\neg r \vee q)$ and $\Box(\neg p \vee \neg q \vee r)$ are obtained;
- $\Box\big(r \leftrightarrow (p \vee q)\big) \equiv \Box\Big(\big(r \supset (p \vee q)\big) \wedge \big((p \vee q) \supset r\big)\Big) \equiv \Box\Big(\big(\neg r \vee p \vee q\big) \wedge \big(\neg(p \vee q) \vee r\big)\Big) \equiv \Box\Big(\big(\neg r \vee p \vee q\big) \wedge \big((\neg p \wedge \neg q) \vee r\big)\Big) \equiv \Box\big((\neg r \vee p \vee q) \wedge (\neg p \vee r) \wedge (\neg q \vee r)\big) \equiv \Box(\neg r \vee p \vee q) \wedge \Box(\neg p \vee r) \wedge \Box(\neg q \vee r)$ — formulas $\Box(\neg r \vee p \vee q)$, $\Box(\neg p \vee r)$ and $\Box(\neg q \vee r)$ are obtained;
- $\Box\big(r \leftrightarrow (p \supset q)\big) \equiv \Box\Big(\big(r \supset (p \supset q)\big) \wedge \big((p \supset q) \supset r\big)\Big) \equiv \Box\Big(\big(\neg r \vee \neg p \vee q\big) \wedge \big(\neg(\neg p \vee q) \vee r\big)\Big) \equiv \Box\Big(\big(\neg r \vee \neg p \vee q\big) \wedge \big((p \wedge \neg q) \vee r\big)\Big) \equiv \Box\big((\neg r \vee \neg p \vee q) \wedge (p \vee r) \wedge (\neg q \vee r)\big) \equiv \Box(\neg r \vee \neg p \vee q) \wedge \Box(p \vee r) \wedge \Box(\neg q \vee r)$ — formulas $\Box(\neg r \vee \neg p \vee q)$, $\Box(p \vee r)$ and $\Box(\neg q \vee r)$ are obtained;
- $\Box(r \leftrightarrow \Box p) \equiv \Box\big((r \supset \Box p) \wedge (\Box p \supset r)\big) \equiv \Box\big((\neg r \vee \Box p) \wedge (\neg \Box p \vee r)\big) \equiv \Box(\neg r \vee \Box p) \wedge \Box(\Diamond \neg p \vee r)$ — formulas $\Box(\neg r \vee \Box p)$ and $\Box(\Diamond \neg p \vee r)$ are obtained;
- $\Box(r \leftrightarrow \Diamond p) \equiv \Box\big((r \supset \Diamond p) \wedge (\Diamond p \supset r)\big) \equiv \Box\big((\neg r \vee \Diamond p) \wedge (\neg \Diamond p \vee r)\big) \equiv \Box(\neg r \vee \Diamond p) \wedge \Box(\Box \neg p \vee r)$ — formulas $\Box(\neg r \vee \Diamond p)$ and $\Box(\Box \neg p \vee r)$ are obtained;

Such reduction finally leaves only one propositional variable in succedent. Let's call it v. Then instead of v in succedent, $\neg v$ can be included into antecedent to obtain sequent of the form $\Box G_1, \Box G_2, \ldots, \Box G_n, l \to$, where $G_i, i \in [1, n]$ are modal disjuncts and l is propositional literal. This sequent is derivable in sequent calculus *GS4* iff $\to F$ is derivable. This theorem was first proved by G. Mints.

Example 4.6.6. Let's transform formula $\neg\Box(p \wedge q) \vee q$:

$$\to \neg\Box(p \wedge q) \vee q$$

$$\Box\Big(r \leftrightarrow (p \wedge q)\Big) \rightarrow \neg\Box r \vee q$$
$$\Box\Big(r \leftrightarrow (p \wedge q)\Big), \Box(u \leftrightarrow \Box r) \rightarrow \neg u \vee q$$
$$\Box\Big(r \leftrightarrow (p \wedge q)\Big), \Box(u \leftrightarrow \Box r), \Box(v \leftrightarrow \neg u) \rightarrow v \vee q$$
$$\Box\Big(r \leftrightarrow (p \wedge q)\Big), \Box(u \leftrightarrow \Box r), \Box(v \leftrightarrow \neg u), \Box\Big(w \leftrightarrow (v \vee q)\Big) \rightarrow w$$
$$\Box\Big(r \leftrightarrow (p \wedge q)\Big), \Box(u \leftrightarrow \Box r), \Box(v \leftrightarrow \neg u), \Box\Big(w \leftrightarrow (v \vee q)\Big), \neg w \rightarrow$$

Now formulas in antecedent can be changed into the needed form according to the proofs above:

$$\Box(\neg r \vee p), \Box(\neg r \vee q), \Box(\neg p \vee \neg q \vee r), \Box(\neg u \vee \Box r), \Box(\Diamond \neg r \vee u),$$
$$\Box(\neg v \vee \neg u), \Box(v \vee u), \Box(\neg w \vee v \vee q), \Box(\neg v \vee w), \Box(\neg q \vee w), \neg w \rightarrow$$

4.7 Knowledge logics

Logics *S5* is often called *knowledge* or *epistemic* logic (gr. επιστημη, *epistēmē* — knowledge). Notation $\Box F$ is read as *agent knows that* F. Axioms *(T)*, *(4)* and *(5)* are interpreted as follows:

- Axiom *(T)*: $\Box F \supset F$ — knowledge or truth axiom ([1]). If agent knows that F, then F is true. I.e. all the knowledge of the agent is correct. This axiom is used to distinguish knowledge from belief.
- Axiom *(4)*: $\Box F \supset \Box\Box F$ — positive introspection. If agent knows, that F, then he (or it) knows that he knows the fact.
- Axiom *(5)*: $\Box F \supset \Box\Diamond F$ — negative introspection. This formula is equivalent to $\neg\Box F \supset \Box\neg\Box F$. If agent does not know if F is true, then he knows that he doesn't know the fact. If the knowledge of humans is discussed, it is very likely that this axiom doesn't hold ([11]), however it usually holds if knowledge of machines is analysed.

In 1957 S. Kanger presented indexed sequent calculus for knowledge logic. Indexes are natural numbers and they are applied to propositional variables. If index $n \in \mathbb{N}$ is applied to propositional variable p, then it is written as the top right index: p^n. Notation F^n means that every propositional variable in F is indexed with number n. E.g. if $F = \Box\Big(p \supset (q \vee \Box p)\Big)$, then $F^2 = \Box\Big(p^2 \supset (q^2 \vee \Box p^2)\Big)$.

Definition 4.7.1. Sequent calculus for epistemic modal knowledge logic *S5* (denoted *GS5*) is obtained from *GPC* by adding two modal rules:

$$\frac{F^m, \Box F^n, \Gamma \rightarrow \Delta}{\Box F^n, \Gamma \rightarrow \Delta}\,(\Box \rightarrow) \qquad \frac{\Gamma \rightarrow \Delta, F^k}{\Gamma \rightarrow \Delta, \Box F^n}\,(\rightarrow \Box)$$

Here m is any natural number, k is a natural number such that the conclusion doesn't contain any propositional variable indexed by k that occur outside the scope of modal operators.

Theorem 4.7.2 (Kanger)**.** *Formula F is valid in knowledge logic S5 iff sequent $\rightarrow F^1$ is derivable in the indexed sequence calculus.*

Example 4.7.3. Let's derive formula $\Box p \supset \Diamond\Box p$.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\Box\neg\Box p^1, \Box p^1 \rightarrow \Box p^1}{\neg\Box p^1, \Box\neg\Box p^1, \Box p^1 \rightarrow}(\neg\rightarrow)}{\Box\neg\Box p^1, \Box p^1 \rightarrow}(\Box\rightarrow)}{\Box p^1 \rightarrow \neg\Box\neg\Box p^1}(\rightarrow\neg)}{\rightarrow \Box p^1 \supset \neg\Box\neg\Box p^1}(\rightarrow\supset)$$

Example 4.7.4. Let's derive formula $\Diamond\Box p \supset \Box p$.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{p^3, \Box p^2 \rightarrow p^3}{\Box p^2 \rightarrow p^3}(\Box\rightarrow)}{\Box p^2 \rightarrow \Box p^1}(\rightarrow\Box)}{\rightarrow \Box p^1, \neg\Box p^2}(\rightarrow\neg)}{\rightarrow \Box p^1, \Box\neg\Box p^1}(\rightarrow\Box)}{\neg\Box\neg\Box p^1 \rightarrow \Box p^1}(\neg\rightarrow)}{\rightarrow \neg\Box\neg\Box p^1 \supset \Box p^1}(\rightarrow\supset)$$

Examples 4.7.3 and 4.7.4 proves that in epistemic logic $\Box p \equiv \Diamond\Box p$. Among other equivalences of *S5* there are:

- $\Box\Diamond F \equiv \Diamond F$;
- $\Box\Diamond\Box F \equiv \Diamond\Box F$;
- $\Diamond\Box\Diamond F \equiv \Box\Diamond F$.

4.8 Multimodal logic

In modal logic one pair of modality operators $\Box$ and $\Diamond$ is analysed. Multimodal logics include a finite number n of such pairs: $\Box_1$, $\Diamond_1$, $\Box_2$, $\Diamond_2$,..., $\Box_n$, $\Diamond_n$. If every modality respects the requirements of monomodal logic K (T, $K4$, $S4$,...), then a multimodal logic is called K_n (respectively T_n, $K4_n$, $S4_n$,...).

Definition 4.8.1. A *multimodal formula* is defined as follows:

- propositional variable is a formula; it is also called *atomic formula*;
- if F is a formula, then $\neg F$, $\Box_k F$ and $\Diamond_k F$ ($k \in [1, n]$) are also formulas;
- if F and G are formulas, then $(F \wedge G)$, $(F \vee G)$, $(F \supset G)$ and $(F \leftrightarrow G)$ are also formulas.

Example 4.8.2. Some formulas of multimodal logic: $\Box_1\Diamond_2\Box_3(p \supset \Diamond_1 q)$, $\neg p \supset (\Diamond_2 p \vee \Box_2 q)$.

To interpret formulas of multimodal logic Kripke structure must contain one relation for each pair of modal operators. Thus in total n relations must be included.

Definition 4.8.3. A *Kripke structure* of multimodal formula F is a multiple $\mathcal{S} = \langle \mathcal{W}, \mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_n, \nu \rangle$, where:

- $\mathcal{W} \neq \varnothing$ is some set, called the set of possible worlds.
- $\mathcal{R}_i, i \in [1, n]$ is a binary relation between elements of set $\mathcal{W}$ for modalities $\Box_i$ and $\Diamond_i$.
- ν is an interpretation function $\nu : \mathcal{P} \times \mathcal{W} \to \{\top, \bot\}$, where $\mathcal{P}$ is a set of propositional variables of formula F.

Hilbert-type and sequent calculi for multimodal logics are obtained from respective calculi for monomodal logics. Here only calculi for K_n are presented.

Definition 4.8.4. Hilbert-type calculus for multimodal logics K_n (denoted HK_n) consists of the same axioms and rules as *HPC*, multimodal Necessity Generalisation rule (NG_l):

$$\frac{F}{\Box_l F}$$

And axiom *(K_l)*: $\Box_l(F \supset G) \supset (\Box_l F \supset \Box_l G)$.

Definition 4.8.5. Gentzen-type calculus for multimodal logics K_n (denoted GK_n) is obtained from *GPC* by adding modal rule:

$$\frac{\Gamma_2 \to F}{\Gamma_1, \Box_l \Gamma_2 \to \Delta, \Box_l F} \, (\to \Box_l)$$

It is also possible to transform multimodal logic formula into predicate logic. More precisely, if F is a formula of multimodal logic K_n, then it is possible to find a formula of predicate logic $\mathrm{Tr}(F)_x$ with one free variable x such that F is satisfiable in K_n iff $\mathrm{Tr}(F)_x$ is satisfiable in predicate logic.

$[F]_\tau$ is defined in a following way:

- $\mathrm{Tr}(G)_x = P(x)$, if G is a propositional variable, here P is predicate variable in which propositional variable G is transformed;
- $\mathrm{Tr}(\neg G)_x = \neg \mathrm{Tr}(G)_x$;
- $\mathrm{Tr}(G \wedge H)_x = \mathrm{Tr}(G)_x \wedge \mathrm{Tr}(H)_x$;
- $\mathrm{Tr}(G \vee H)_x = \mathrm{Tr}(G)_x \vee \mathrm{Tr}(H)_x$;
- $\mathrm{Tr}(G \supset H)_x = \mathrm{Tr}(G)_x \supset \mathrm{Tr}(H)_x$;
- $\mathrm{Tr}(\Box_l G)_x = \forall y \Big(R_l(x, y) \supset \mathrm{Tr}(G)_y \Big)$;
- $\mathrm{Tr}(\Diamond_l G)_x = \exists y \Big(R_l(x, y) \wedge \mathrm{Tr}(G)_y \Big)$.

Here $k \in [i, n]$, y is a new individual variable. Analogously as described in Section 4.5 formulas of other multimodal logics can be transformed into predicate logic.

Chapter 5

Temporal logics

5.1 Branching time logics

Temporal logics analyse formulas over time. In different logics such concepts as *always in the future*, *next time* or *until* can be analysed. There is a wide variety of temporal logics. Most of them are:

- propositional or first-order;
- finite or infinite;
- linear or branching time;
- discrete or continuous;
- oriented to the future or to the past and the future.

Let's start with logic *CTL* (Computation Tree Logic), which is one of *branching time* logics. This logic combines quantifiers $\forall$ and $\exists$ together with modal operators $\Box$ and $\Diamond$. Branching times means that future is modelled as a tree. The branches of the tree represent different possible scenarios of the future. Thus quantifiers define the situation in different branches: $\forall$ means *in every branch* and $\exists$ means *there is a branch*. Modal operators define the future in one branch: $\Box$ means *always in the future* and $\Diamond$ means *at some time in the future*. Moreover, two temporal operators are used: $\circ$ means *in the next moment of time* and $\mathcal{U}$ means *until*.

Definition 5.1.1. A *CTL formula* is defined in a recursive way:

- $\top$ and $\bot$ are formulas;
- propositional variable is a formula; it is also called *atomic formula*;
- if F is a formula, then $\neg F$ is also a formula;
- if F and G are formulas, then $(F \wedge G)$, $(F \vee G)$, $(F \supset G)$ and $(F \leftrightarrow G)$ are also formulas;
- if F is formula, then $\circ F$, $\Diamond F$ and $\Box F$ are auxiliary formulas;
- if F and G are formulas, then $(F \,\mathcal{U}\, G)$ is auxiliary formula;

- if F is auxiliary formula, then $\forall F$ and $\exists F$ are formulas.

As the definition states, expressions $\Box\Box p$ and $\Diamond\Diamond p$ are neither formulas nor auxiliary formulas.

Example 5.1.2. Several formula examples: $\forall\Box\exists\circ\forall(\neg p \,\mathcal{U}\, q)$, $\forall\Box(p \supset q)$, $\exists\Box p$, $\exists\Box(p \vee \exists\Box q)$.

To interpret formulas of *CTL* a modified Kripke structure is used. Let's define it:

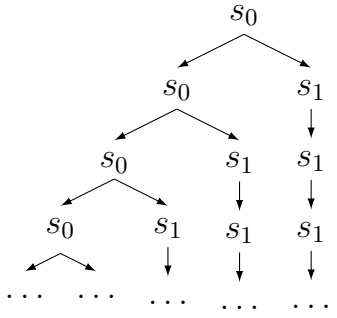
Definition 5.1.3. A *Kripke structure* for *CTL* formula F is a quadruple $\langle \mathcal{W}, \mathcal{I}, \mathcal{R}, \nu \rangle$, where:

- $\mathcal{W} \neq \varnothing$ is a set of states;
- $\mathcal{I} \subseteq \mathcal{W}, \mathcal{I} \neq \varnothing$ is a set of starting states;
- $\mathcal{R}$ is a relation in set $\mathcal{W}$, called transition;
- $\nu : \mathcal{P} \times \mathcal{W} \to \{\top, \bot\}$, where $\mathcal{P}$ is a set of propositional variables of F, is an interpretation function.

It must be noted, that usually only finite Kripke structures are analysed ($\mathcal{W}$ must be finite). Moreover, it is usually required that the relation $\mathcal{R}$ be complete. A finite sequence of states $s_0, s_1, \ldots, s_n$ is called a *path* from s_0 to s_n, if $s_i \mathcal{R} s_{i+1}, i \in [0, n-1]$.

Example 5.1.4. Suppose Kripke structure $\langle \mathcal{W}, \mathcal{I}, \mathcal{R}, \nu \rangle$ is defined as follows: $\mathcal{W} = \{s_0, s_1\}$, $\mathcal{I} = \{s_0\}$, $\mathcal{R} = \{(s_0, s_0), (s_0, s_1), (s_1, s_1)\}$ and ν is some interpretation. Such structure can be represented using graph:

All the possible paths from starting state s_0 can be demonstrated using infinite tree:



Now let's define how the formula is interpreted using Kripke structure. The fact that formula F is true in state s of Kripke structure $\mathcal{S}$ is denoted (as usual) $\mathcal{S}, s \vDash F$. To state that auxiliary formula G is true in path π of Kripke structure $\mathcal{S}$ a similar notation is used: $\mathcal{S}, \pi \vDash G$.

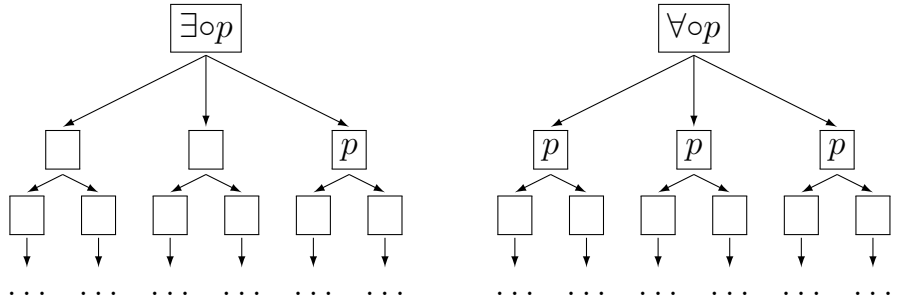
Definition 5.1.5. Suppose $\mathcal{S} = \langle \mathcal{W}, \mathcal{I}, \mathcal{R}, \nu \rangle$ is a Kripke structure and $s \in \mathcal{W}$ is some state. The definition of $\mathcal{S}, s \vDash F$ is by induction on the structure of F:

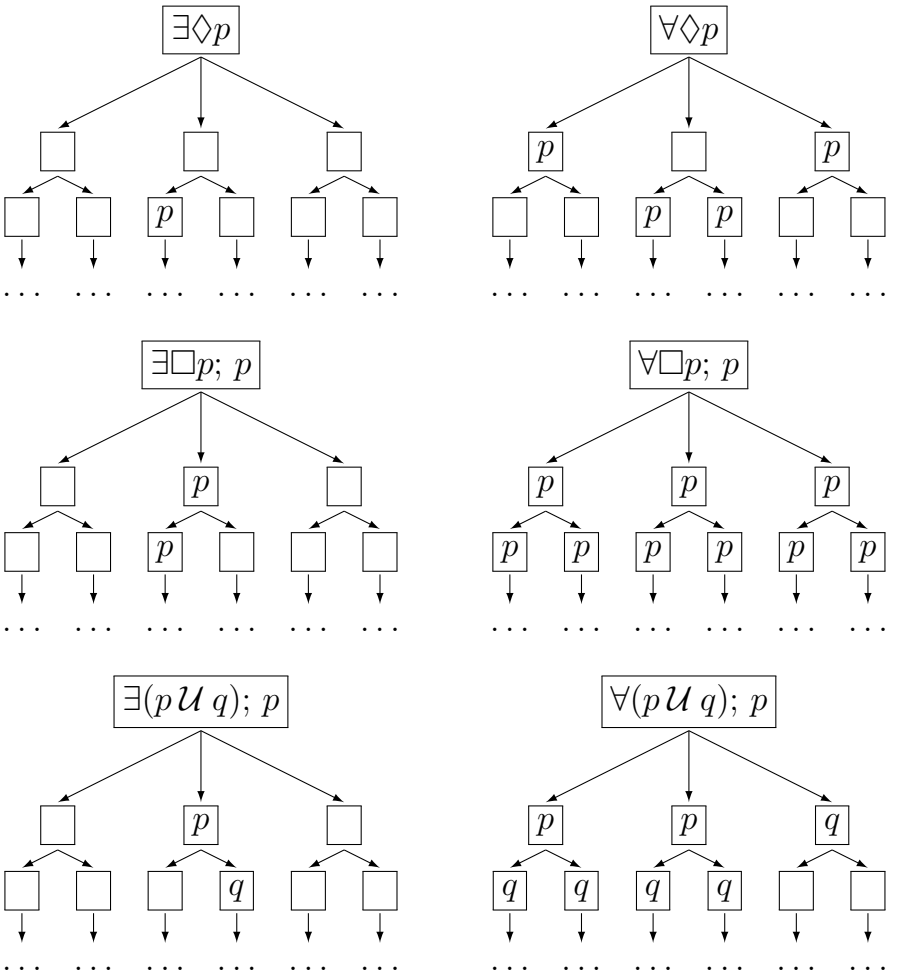
- if $F = \top$, then $\mathcal{S}, s \vDash F$;
- if $F = \bot$, then $\mathcal{S}, s \nvDash F$;
- if F is a propositional variable, then $\mathcal{S}, s \vDash F$, iff $\nu(F, s) = \top$;
- if $F = G \wedge H$, then $\mathcal{S}, s \vDash F$, iff $\mathcal{S}, s \vDash G$ and $\mathcal{S}, s \vDash H$. The cases when $F = \neg G$, $F = G \vee H$, $F = G \supset H$ and $F = G \leftrightarrow H$ are defined analogously (see e.g. Definition 4.1.5);
- if $F = \forall G$, then $\mathcal{S}, s \vDash F$, iff for any path π from s it is true that $\mathcal{S}, \pi \vDash G$;
- if $F = \exists G$, then $\mathcal{S}, s \vDash F$, iff there is a path π from s such that $\mathcal{S}, \pi \vDash G$;
- if $F = \circ G$ and $\pi = s_0, s_1, \ldots$, then $\mathcal{S}, \pi \vDash F$, iff $\mathcal{S}, s_1 \vDash G$;
- if $F = \Diamond G$ and $\pi = s_0, s_1, \ldots$, then $\mathcal{S}, \pi \vDash F$, iff there is $i \in [0, \infty)$ such that $\mathcal{S}, s_i \vDash G$;
- if $F = \Box G$ and $\pi = s_0, s_1, \ldots$, then $\mathcal{S}, \pi \vDash \Box F$, iff $\mathcal{S}, s_i \vDash G$ for any $i \in [0, \infty)$;
- if $F = G \,\mathcal{U}\, H$ and $\pi = s_0, s_1, \ldots$, then $\mathcal{S}, \pi \vDash F$, iff there is $i \in [0, \infty)$ such that $\mathcal{S}, s_i \vDash H$ and for any $j \in [0, i)$ it is true that $\mathcal{S}, s_j \vDash G$;

Thus now the meaning of quantifiers and temporal operators can be explained in more detail:

- $\exists\circ$ — there is a path such that in its next state formula is true;
- $\forall\circ$ — for any path in its next state formula is true;
- $\exists\Diamond$ — there is a path which contains a state in which formula is true;
- $\forall\Diamond$ — in any path there is a state in which formula is true;
- $\exists\Box$ — there is a path such that in its every state formula is true;
- $\forall\Box$ — for any path in every state formula is true;
- $\exists\mathcal{U}$ — there is a path such that until some state formula is true;
- $\forall\mathcal{U}$ — for any path there is a state until which formula is true;

These situations can be illustrated by following diagrams:





Definition 5.1.6. It is said that formula F is *true* in Kripke structure $\mathcal{S} = \langle \mathcal{W}, \mathcal{I}, \mathcal{R}, \nu \rangle$ (denoted $\mathcal{S} \vDash F$) if for any starting state $s \in \mathcal{I}$ it is true that $\mathcal{S}, s \vDash F$.

Definition 5.1.7. It is said that formulas F and G are *equivalent* if for any Kripke structure $\mathcal{S}$ it is true that $\mathcal{S} \vDash F$ iff $\mathcal{S} \vDash G$.

Logic *CTL* is decidable.

5.2 Linear Temporal logic

5.2.1 Syntax and semantics

Propositional Linear Temporal Logic (denoted *PLTL*) is obtained by enriching propositional logic with temporal operators $\circ$, $\Diamond$, $\Box$ and $\mathcal{U}$.

Definition 5.2.1. A *PLTL formula* is defined in a recursive way:

- $\top$ and $\bot$ are formulas;
- propositional variable is a formula; it is also called *atomic formula*;
- if F is a formula, then $\neg F$, $\circ F$, $\Box F$ and $\Diamond F$ are also formulas;

- if F and G are formulas, then $(F \vee G)$, $(F \wedge G)$, $(F \supset G)$, $(F \leftrightarrow G)$ and $(F \,\mathcal{U}\, G)$ are also formulas.

Example 5.2.2. The following expressions are *PLTL* formulas: $\Box(p \vee \circ q)$, $\Box\circ\big(\Diamond\neg q \supset (p \,\mathcal{U}\, q)\big)$, $\neg\Diamond p \supset \circ\Diamond\neg p$.

In contrast to branching time, linear time means that time doesn't branch and only one possible future is analysed. Thus it is interpreted using a Kripke structure with discrete linear (every element has exactly one next element), reflexive and transitive relation. More precisely:

Definition 5.2.3. A *Kripke structure* of *PLTL* formula F is a pair $\langle \mathcal{W}, \nu \rangle$, where:

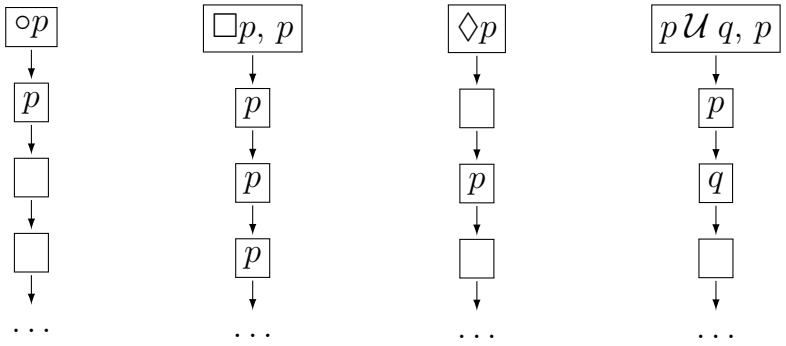
- $\mathcal{W}$ is a sequence of states $s_0, s_1, s_2, \ldots$;
- $\nu : \mathcal{P} \times \mathcal{W} \to \{\top, \bot\}$ is an interpretation function, where $\mathcal{P}$ is a set of propositional variables of F.

Traditionally the fact that formula F is true in state s_i of Kripke structure $\mathcal{S}$ is denoted as $\mathcal{S}, s_i \vDash F$.

Definition 5.2.4. Suppose $\mathcal{S} = \langle \mathcal{W}, \nu \rangle$ is a Kripke structure, where sequence $\mathcal{W} = s_0, s_1, \ldots$ and $s_i \in \mathcal{W}$ is some state. The definition of $\mathcal{S}, s_i \vDash F$ is by induction on the structure of F:

- if $F = \top$, then $\mathcal{S}, s_i \vDash F$;
- if $F = \bot$, then $\mathcal{S}, s_i \nvDash F$;
- if F is a propositional variable, then $\mathcal{S}, s_i \vDash F$, iff $\nu(p, s_i) = \top$;
- if $F = \neg G$, then $\mathcal{S}, s_i \vDash F$, iff $\mathcal{S}, s_i \nvDash G$;
- if $F = G \wedge H$, then $\mathcal{S}, s_i \vDash F$, iff $\mathcal{S}, s_i \vDash G$ and $\mathcal{S}, s_i \vDash H$;
- if $F = G \vee H$, then $\mathcal{S}, s_i \nvDash F$, iff $\mathcal{S}, s_i \nvDash G$ and $\mathcal{S}, s_i \nvDash H$;
- if $F = G \supset H$, then $\mathcal{S}, s_i \nvDash F$, iff $\mathcal{S}, s_i \vDash G$ and $\mathcal{S}, s_i \nvDash H$;
- if $F = G \leftrightarrow H$, then $\mathcal{S}, s_i \nvDash F$, iff $\mathcal{S}, s_i \vDash G$ and $\mathcal{S}, s_i \vDash H$ or $\mathcal{S}, s_i \nvDash G$ and $\mathcal{S}, s_i \nvDash H$;
- if $F = \circ G$, then $\mathcal{S}, s_i \vDash F$, iff $\mathcal{S}, s_{i+1} \vDash G$;
- if $F = \Box G$, then $\mathcal{S}, s_i \vDash F$, iff for every $j \geqslant i$ it is true that $\mathcal{S}, s_j \vDash G$;
- if $F = \Diamond G$, then $\mathcal{S}, s_i \vDash F$, iff there is $j \geqslant i$ such that $\mathcal{S}, s_j \vDash G$;
- if $F = G \,\mathcal{U}\, H$, then $\mathcal{S}, s_i \vDash F$, iff there is $j \geqslant i$ such that $\mathcal{S}, s_j \vDash H$ and for every $k \in [i, j)$ it is true that $\mathcal{S}, s_k \vDash G$;

Semantics of temporal operators $\circ$, $\Box$, $\Diamond$ and $\mathcal{U}$ can be explained using these schemes:



It is said that formula of linear temporal logic F is *satisfiable* if there is a Kripke structure $\mathcal{S}$ with state s_i such that $\mathcal{S}, s_i \vDash F$. It is said that formula of linear temporal logic is *valid* (denoted $\vDash F$) if it is true in any state of any interpretation.

Example 5.2.5. Formula $\Box\Diamond q$ is satisfiable in *PLTL*. To show that it is enough to find a Kripke structure with infinite set of states in which q is true. It can be noted that sequence of states in every Kripke structure is the same, thus it is enough to define an interpretation function. One such interpretation function is: $\nu(q, s_i) = \top$ iff i is a prime number. It is easy to see that formula $\Box\Diamond q$ is true in Kripke structure $\langle \mathcal{W}, \nu \rangle$, where $\mathcal{W} = s_0, s_1, \ldots$.

Example 5.2.6. Formula $\Box\big(p \supset \circ(q\,\mathcal{U}\,r)\big)$ is satisfiable in *PLTL*. To show that it is enough to find a Kripke structure, which has such property: if p is true in some state s_i, then starting next state s_{i+1} variable q must be true in all the states up to some state $s_j, j > i$, in which r is true. Of course the simplest solution is to define an interpretation function in a following way: $\nu(p, s_i) = \bot$ for any state s_i of the Kripke structure. It doesn't matter what is the interpretation of propositional variables q and r.

The following formulas are valid in *PLTL*:

1. $\Box F \supset F$;
2. $\Box F \leftrightarrow \Box\Box F$;
3. $\circ F \leftrightarrow \neg\circ\neg F$;

Here F is any formula of linear temporal logics. Formula 1 describes reflexivity of the structure, formula 2 — transitivity and formula 3 — linearity.

Sometimes in linear temporal logics another operator $\mathcal{R}$, called *release*, is also used. This operator is dual to operator until: $F\,\mathcal{R}\,G \equiv \neg(\neg F\,\mathcal{U}\,\neg G)$. Moreover, $\Diamond F \equiv \top\,\mathcal{U}\,F$ and $\Box F \equiv \bot\,\mathcal{R}\,F$. Thus for any formula of linear temporal logic it is possible to find an equivalent formula, in which only operators $\neg$, $\vee$, $\circ$ and $\mathcal{U}$ occur.

5.2.2 Hilbert-type calculus

Definition 5.2.7. Hilbert-type calculus for linear temporal logics (denoted $HPLTL_o$) consists of the same axioms and rules as *HPC*, Necessity Generalisation rule (*NG*):

$$\frac{F}{\Box F}$$

And axioms for temporal operators:

5.1. $\Box(F \supset G) \supset (\Box F \supset \Box G)$;

5.2. $\circ\neg F \supset \neg\circ F$;

5.3. $\neg\circ F \supset \circ\neg F$;

5.4. $\circ(F \supset G) \supset (\circ F \supset \circ G)$;

5.5. $\Box F \supset (F \wedge \circ\Box F)$;

5.6. $\Box(F \supset \circ F) \supset (F \supset \Box F)$;

5.7. $(F\,\mathcal{U}\,G) \supset \Diamond G$;

5.8. $(F\,\mathcal{U}\,G) \supset \Big(G \vee \big(F \wedge \circ(F\,\mathcal{U}\,G)\big)\Big)$;

5.9. $\Big(G \vee \big(F \wedge \circ(F\,\mathcal{U}\,G)\big)\Big) \supset (F\,\mathcal{U}\,G)$;

D. Gabbay in 1980 proved the soundness and completeness of this calculus.

In linear temporal logic as well as in modal logic the following statement is true:

Lemma 5.2.8. *If $F(p_1, p_2, \ldots, p_n)$ is valid formula of propositional logic, $G_1, G_2, \ldots, G_n$ are some formulas of linear temporal logic, then formula $F(G_1, G_2, \ldots, G_n)$ is valid in linear temporal logic.*

Example 5.2.9. $\neg\neg p \supset p$ is valid formula of propositional logic, therefore $\neg\neg\Box\Diamond q \supset \Box\Diamond q$ is valid formula of linear temporal logic.

Formula $p \supset (q \supset p)$ is valid in propositional logic, therefore formula $(\Diamond p \supset \Box q) \supset \Big(\circ q \supset (\Diamond p \supset \Box q)\Big)$ is valid in *PLTL*.

Derivation search in calculus $HPLTL_o$ is quite difficult, therefore in practice usually a similar calculus (denoted $HPLTL_1$) is used, which composes of axioms *5.1* — *5.9*, rules *MP*, *NG* and a new rule, called substitution (*SUB*):

$$\frac{F(p_1, p_2, \ldots, p_n)}{F(G_1, G_2, \ldots, G_n)}$$

In this rule $F(p_1, p_2, \ldots, p_n)$ is some valid formula of propositional logics and $G_1, G_2, \ldots, G_n$ — are some formulas or linear temporal logic.

For convenience let's include three more rules into calculus:

$$\frac{F \supset G}{\Box F \supset \Box G} \qquad \frac{F \supset G}{\circ F \supset \circ G} \qquad \frac{F \supset \circ F}{F \supset \Box F}$$

Let's name these additional rules *AD1*, *AD2* and *AD3* respectively and let's denote the resulting calculus $HPLTL_2$. It is easy to prove that these rules are admissible in calculus $HPLTL_1$, thus every formula is derivable in $HPLTL_2$ iff it is derivable in $HPLTL_1$.

Let's show some examples of derivation in calculus $HPLTL_2$.

Example 5.2.10. Derivation of formula $\Box p \supset \Box\Box p$:

1.	$\big(p \supset (q \wedge r)\big) \supset (p \supset r)$	Valid formula of propositional logic.
2.	$\big(\Box p \supset (p \wedge \circ\Box p)\big) \supset (\Box p \supset \circ\Box p)$	*SUB* rule from 1, $\{^{\Box p}/_p, {}^{p}/_q, {}^{\circ\Box p}/_r\}$.
3.	$\Box p \supset (p \wedge \circ\Box p)$	Axiom *5.5*, $\{^{p}/_F\}$.
4.	$\Box p \supset \circ\Box p$	*MP* rule from 3 and 2.
5.	$\Box p \supset \Box\Box p$	*AD3* rule from 4.

Example 5.2.11. Derivation of formula $\circ p \supset \neg\circ\neg p$:

1.	$(p \supset q) \supset (\neg q \supset \neg p)$	Valid formula of propositional logic.
2.	$(\circ\neg p \supset \neg\circ p) \supset (\neg\neg\circ p \supset \neg\circ\neg p)$	*SUB* rule from 1, $\{^{\circ\neg p}/_p, {}^{\neg\circ p}/_q\}$.
3.	$\circ\neg p \supset \neg\circ p$	Axiom *5.2*, $\{^{p}/_F\}$.
4.	$\neg\neg\circ p \supset \neg\circ\neg p$	*MP* rule from 3 and 2.
5.	$(\neg\neg p \supset q) \supset (p \supset q)$	Valid formula of propositional logic.
6.	$(\neg\neg\circ p \supset \neg\circ\neg p) \supset (\circ p \supset \neg\circ\neg p)$	*SUB* rule from 5, $\{^{\circ p}/_p, {}^{\neg\circ\neg p}/_q\}$.
7.	$\circ p \supset \neg\circ\neg p$	*MP* rule from 4 and 6.

Some equivalences of *PLTL* are:

- $\circ(F \wedge G) \equiv \circ F \wedge \circ G$;
- $\neg\circ F \equiv \circ\neg F$;
- $\neg\Box F \equiv \Diamond\neg F$;
- $\Box F \equiv F \wedge \circ\Box F$;
- $\Diamond F \equiv F \vee \circ\Diamond F$;
- $F \,\mathcal{U}\, G \equiv G \vee \big(F \wedge \circ(F \,\mathcal{U}\, G)\big)$.

5.2.3 Sequent calculus

In 2007 J. Gaintzarian, M. Hermo and others defined a sequent calculus without cut [3]. This calculus can be applied to formulas, that do not contain logical operators $\wedge$, $\Box$, $\Diamond$ and logical constant $\top$. However, every formula can be transformed into such form using following equivalences:

- $F \wedge G \equiv \neg(\neg F \vee \neg G)$;
- $\Box F \equiv \neg\Diamond\neg F$;

- $\Diamond F \equiv \top \mathcal{U} F$;
- $\top \equiv \neg\bot$.

Furthermore, succedent of every sequent in this calculus consist of exactly one formula.

Definition 5.2.12. A sequent calculus for *PLTL* (denoted *GPLTL*) consists of axiom $F, \Gamma \to F$ and following rules:

Logical rules:

$$\frac{\Gamma \to F}{\neg F, \Gamma \to H}\,(\neg\to) \qquad \frac{F, \Gamma \to \bot}{\Gamma \to \neg F}\,(\to\neg)$$

$$\frac{F, \Gamma \to H \qquad G, \Gamma \to H}{F \vee G, \Gamma \to H}\,(\vee\to)$$

$$\frac{\Gamma \to F}{\Gamma \to F \vee G}\,(\to\vee)_1 \qquad \frac{\Gamma \to G}{\Gamma \to F \vee G}\,(\to\vee)_2$$

Temporal rules:

$$\frac{\Gamma_1 \to F}{\circ\Gamma_1, \Gamma_2 \to \circ F}\,(\to\circ) \qquad \frac{\circ\neg F, \Gamma \to H}{\neg\circ F, \Gamma \to H}\,(\neg\circ\to) \qquad \frac{\Gamma \to \neg\circ F}{\Gamma \to \circ\neg F}\,(\to\circ\neg)$$

$$\frac{G, \Gamma \to H \qquad F, \neg G, \circ(F \mathcal{U} G), \Gamma \to H}{F \mathcal{U} G, \Gamma \to H}\,(\mathcal{U}\to)_1$$

$$\frac{G, \Gamma \to H \qquad F, \neg G, \circ\Big(\big(F \wedge (\neg\Gamma^{\vee} \vee H)\big) \mathcal{U} G\Big), \Gamma \to H}{F \mathcal{U} G, \Gamma \to H}\,(\mathcal{U}\to)_2$$

Here $\neg\Gamma^{\vee}$ is a disjunction of the negations of formulas from Γ. E.g. if $\Gamma = F_1, F_2, \ldots, F_n$, then $\neg\Gamma^{\vee} = \neg F_1 \vee \neg F_2 \vee \ldots \vee \neg F_n$. If $\Gamma = \varnothing$, then $\neg\Gamma^{\vee} = \bot$.

$$\frac{\neg F, \Gamma \to G \qquad F, \neg\circ(F \mathcal{U} G), \Gamma \to G}{\Gamma \to F \mathcal{U} G}\,(\to\mathcal{U})$$

Structural rules:

$$\frac{\Gamma_1 \to H}{\Gamma_1, \Gamma_2 \to H}\,(w\to) \qquad \frac{\neg F, \Gamma \to \bot}{\Gamma \to F}\,(\to t) \qquad \frac{\Gamma \to \circ\bot}{\Gamma \to F}\,(\circ\bot)$$

Rule $(w\to)$ is called weakening and rule $(\to t)$ is called transfer.

For convenience, sequents $\Gamma, F, \neg F \to H$ and $\Gamma, \bot \to H$ are also considered as axioms. These sequents are derivable in *GPLTL*.

The following rules are also admissible in *GPLTL*:

$$\frac{\neg G, \Gamma \to F}{\neg F, \Gamma \to G} \qquad \frac{G, \Gamma \to F}{\neg F, \Gamma \to \neg G} \qquad \frac{F, \Gamma \to H}{\neg\neg F, \Gamma \to H} \qquad \frac{\Gamma \to \bot}{\circ\Gamma \to F}$$

5.3 Finite linear temporal logic

5.3.1 Syntax and semantics

In finite linear temporal logic, contrary to *PLTL*, time is finite. That is, Kripke structure contains only finite number of states. However, both present and past is analysed. Together with temporal operators of *PLTL* ($\Box$ — *always in the future*, $\Diamond$ — *sometime in the future*, $\circ$ — *in the next moment of time*), new temporal operators are used: $\overleftarrow{\Box}$ — *always in the past*, $\overleftarrow{\Diamond}$ — *sometime in the past*, $\overleftarrow{\circ}$ — *in the previous moment of time*, $\odot$ — *certainly in the next moment of time*, $\overleftarrow{\odot}$ — *certainly in the previous moment of time*.

Definition 5.3.1. *Formula* of finite linear temporal logic is defined as follows:

propositional variable is a formula; it is also called *atomic formula*;

if F is a formula, then $\neg F$, $\circ F$, $\overleftarrow{\circ} F$, $\odot F$, $\overleftarrow{\odot} F$, $\Box F$, $\overleftarrow{\Box} F$, $\Diamond F$ and $\overleftarrow{\Diamond} F$ are also formulas;

if F and G are formulas, then $(F \vee G)$ and $(F \wedge G)$ are also formulas

As it can be noticed from the definition, operation $\supset$ and $\leftrightarrow$ are not used. They can be replaced by $\neg$, $\wedge$ and $\vee$. Moreover, usually just formulas in which negation occurs only in front of propositional variables are analysed. Such form can always be obtained using the following equivalences:

$$\neg\circ F \equiv \odot\neg F \quad \neg\odot F \equiv \circ\neg F \quad \neg\Box F \equiv \Diamond\neg F \quad \neg\Diamond F \equiv \Box\neg F$$
$$\neg\overleftarrow{\circ} F \equiv \overleftarrow{\odot}\neg F \quad \neg\overleftarrow{\odot} F \equiv \overleftarrow{\circ}\neg F \quad \neg\overleftarrow{\Box} F \equiv \overleftarrow{\Diamond}\neg F \quad \neg\overleftarrow{\Diamond} F \equiv \overleftarrow{\Box}\neg F$$

As it was mentioned earlier, time is finite in such logic. To define time, a *time structure* is used. A time structure is a finite sequence of states $s_0, s_1, \ldots, s_n$. For convenience, notation $s_0 < s_1 < \cdots < s_n$ and $s_i + k = s_{i+k}$, where $i \in [0, n-k]$ is used. Similarly $s_i - k = s_{i-k}$, where $i \in [k, n]$. A time structure is used in Kripke structure:

Definition 5.3.2. A *Kripke structure* for finite linear temporal logic formula F is a pair $\langle \mathcal{W}, \nu \rangle$, where:

- $\mathcal{W}$ is some time structure;
- $\nu : \mathcal{P} \times \mathcal{W} \to \{\top, \bot\}$, where $\mathcal{P}$ is a set of propositional variables of formula F, is an interpretation function.

The fact that formula F is true in state s of structure $\mathcal{S}$ is denoted $\mathcal{S}, s \vDash F$ and it is defined in a similar way as for other temporal logics:

Definition 5.3.3. Suppose $\mathcal{S} = \langle \mathcal{W}, \nu \rangle$ is a Kripke structure for finite linear temporal logic and $s \in \mathcal{W}$ is some state. A definition of $\mathcal{S}, s \vDash F$ is by induction on the form of formula F:

- if F is a propositional variable, then $\mathcal{S}, s \vDash F$ iff $\nu(F, s) = \top$;
- if $F = \neg G$, $F = G \wedge H$ or $F = G \vee H$, then the definition is analogous to *PLTL* case in Definition 5.2.4;
- if $F = \Box G$, then $\mathcal{S}, s \vDash F$ iff for all $s' \in \mathcal{W}$ such that $s' \geqslant s$ it is true that $\mathcal{S}, s' \vDash G$;
- if $F = \Diamond G$, then $\mathcal{S}, s \vDash F$ iff there is $s' \in \mathcal{W}$ such that $s' \geqslant s$ and $\mathcal{S}, s' \vDash G$;
- if $F = \overleftarrow{\Box} G$, then $\mathcal{S}, s \vDash F$ iff for all $s' \in \mathcal{W}$ such that $s' \leqslant s$ it is true that $\mathcal{S}, s' \vDash G$;
- if $F = \overleftarrow{\Diamond} G$, then $\mathcal{S}, s \vDash F$ iff there is $s' \in \mathcal{W}$ such that $s' \leqslant s$ and $\mathcal{S}, s' \vDash G$;
- if $F = \circ G$, then $\mathcal{S}, s \vDash F$ iff $s + 1 \notin \mathcal{W}$ (i.e. if $\mathcal{W} = s_0, s_1, \ldots, s_n$, then $s = s_n$) or $\mathcal{S}, s + 1 \vDash G$;
- if $F = \odot G$, then $\mathcal{S}, s \vDash F$ iff $s + 1 \in \mathcal{W}$ (i.e. $s \neq s_n$) and $\mathcal{S}, s + 1 \vDash G$;
- if $F = \overleftarrow{\circ} G$, then $\mathcal{S}, s \vDash F$ iff $s - 1 \notin \mathcal{W}$ (i.e. $s = s_0$) or $\mathcal{S}, s - 1 \vDash G$;
- if $F = \overleftarrow{\odot} G$, then $\mathcal{S}, s \vDash F$ iff $s - 1 \in \mathcal{W}$ (i.e. $s \neq s_0$) and $\mathcal{S}, s - 1 \vDash G$.

It is said that formula F is *true* in Kripke structure $\mathcal{S} = \langle \mathcal{W}, \nu \rangle$ (denoted $\mathcal{S} \vDash F$), where $\mathcal{W} = s_0, s_1, \ldots, s_n$, iff F is true in initial state s_0. Formula is *satisfiable*, if there is a Kripke structure in which the formula is true. Formula is *valid*, if it is true in every Kripke structure.

5.3.2 Tableaux calculus

Once again although the method is similar to the ones presented in Sections 2.6 and 4.4, however the expressions used in the method differs. In tableaux calculus for finite linear temporal logic expressions of the form $s : F || \{\delta\}$ are analysed. Here s is a state and δ is restriction of the states. A set of all the restrictions in the branch from root to the current node is denoted as Δ. If an application of the rule does not add a new restriction, then the expression of the form $s : F$ is presented.

Restrictions are used to decide if the rule can be applied and also to check if the branch is closed. Moreover, expressions of the form $\Delta \vdash \delta$ are used. This means that from set of restrictions Δ a restriction δ is derivable. All the restrictions used here are comparisons of two states, including operators

$=$, $<$ and $\leqslant$. Thus the derivability is understood in traditional arithmetic sense. Several examples are:

- if $\delta \in \Delta$, then $\Delta \vdash \delta$;
- $\Delta \vdash s = s$;
- if $\Delta \vdash s_1 < s_2$ and $\Delta \vdash s_2 < s_3$, then also $\Delta \vdash s_1 < s_3$ (the same applies for operators $=$ and $\leqslant$);
- if $\Delta \vdash s_1 = s_2$ or $\Delta \vdash s_1 < s_2$, then also $\Delta \vdash s_1 \leqslant s_2$;
- if $\Delta \vdash s_1 = s_2 + 1$, then $\Delta \vdash s_2 < s_1$.

It is said that set Δ is *contradictory* if:

- $\Delta \vdash s_1 < s_2$ and $\Delta \vdash s_2 = s_1$ or
- $\Delta \vdash s_1 < s_2$ and $\Delta \vdash s_2 < s_1$ or
- $\Delta \vdash s_1 < s_2$ and $\Delta \vdash s_2 \leqslant s_1$.

Definition 5.3.4. Tableaux calculus for finite linear temporal logic (denoted *TFLTL*) contains the following rules:

$$\frac{s : F \wedge G}{\begin{array}{c} s : F \\ s : G \end{array}} \qquad \frac{s : F \vee G}{s : F \mid s : G} \qquad \frac{s : \Box F}{s^+ : F} \qquad \frac{s : \overleftarrow{\Box} F}{s^- : F}$$

The rule to $s : \Box F$ can be applied if $\Delta \vdash s \leqslant s^+$ and the rule to $s : \overleftarrow{\Box} F$ can be applied if $\Delta \vdash s^- \leqslant s$.

$$\frac{s : \Diamond F}{s' : F || \{s \leqslant s'\}} \qquad \frac{s : \circ F}{s' : F || \{s' = s + 1\}} \qquad \frac{s : \odot F}{s' : F || \{s' = s + 1\}}$$

$$\frac{s : \overleftarrow{\Diamond} F}{s' : F || \{s' \leqslant s\}} \qquad \frac{s : \overleftarrow{\circ} F}{s' : F || \{s = s' + 1\}} \qquad \frac{s : \overleftarrow{\odot} F}{s' : F || \{s = s' + 1\}}$$

The rule to $s : \circ F$ can be applied if there is s'' such that $\Delta \vdash s \leqslant s''$ and the rule to $s : \overleftarrow{\circ} F$ can be applied if there is s'' such that $\Delta \vdash s'' \leqslant s$. In all the rules s' is a new state.

$$\frac{s : F}{s : F || \{s < s'\} \mid s : F || \{s = s'\} \mid s : F || \{s' < s\}}$$

This rule is called sync.

Definition 5.3.5. It is said that a branch of a derivation search tree is *closed* if at least one of the following conditions is satisfied:

- set of restrictions Δ is contradictory;
- in the branch there are two expressions $s' : F$ and $s'' : \neg F$ such that $\Delta \vdash s' = s''$;
- in the branch there is an expression $s : F$ such that $\Delta \vdash s < s_0$, where s_0 is the initial state.

If a branch is not closed, then it is *open*.

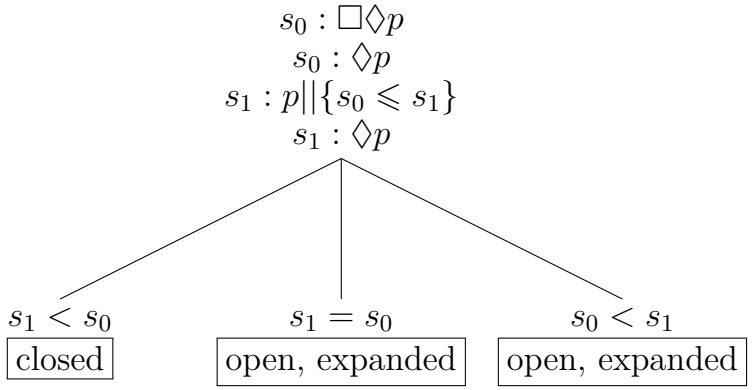
Definition 5.3.6. State s in a branch of a derivation search tree is *expanded* if all the following conditions are satisfied:

- for every expression $s : F \wedge G$ that occurs in the branch and every state s' such that $\Delta \vdash s = s'$ expressions $s' : F$ and $s' : G$ also occur in the branch;
- for every expression $s : F \vee G$ that occurs in the branch and every state s' such that $\Delta \vdash s = s'$ at least one of expressions $s' : F$ or $s' : G$ also occurs in the branch;
- for every expression $s : \Box F$ that occurs in the branch and every state s' such that $\Delta \vdash s \leqslant s'$ expression $s' : F$ also occurs in the branch;
- for every expression $s : \overleftarrow{\Box} F$ that occurs in the branch and every state s' such that $\Delta \vdash s' \leqslant s$ expression $s' : F$ also occurs in the branch;
- for every expression $s : \Diamond F$ there is s' such that $\Delta \vdash s \leqslant s'$ and expression $s' : F$ also occurs in the branch;
- for every expression $s : \overleftarrow{\Diamond} F$ there is s' such that $\Delta \vdash s' \leqslant s$ and expression $s' : F$ also occurs in the branch;
- for every expression $s : \circ F$ if $\Delta \vdash s < s''$ for some s'', then there is s' such that $\Delta \vdash s' = s+1$ and expression $s' : F$ also occurs in the branch;
- for every expression $s : \overleftarrow{\circ} F$ if $\Delta \vdash s'' < s$ for some s'', then there is s' such that $\Delta \vdash s = s'+1$ and expression $s' : F$ also occurs in the branch;
- for every expression $s : \odot F$ there is s' such that $\Delta \vdash s' = s + 1$ and expression $s' : F$ also occurs in the branch;
- for every expression $s : \overleftarrow{\circ} F$ there is s' such that $\Delta \vdash s = s' + 1$ and expression $s' : F$ also occurs in the branch.

A branch is said to be *expanded* if all the states that occur in the branch are expanded and set Δ is fully ordered (i.e. for any two states s_1 and s_2 that are part of Δ either $\Delta \vdash s_1 < s_2$ or $\Delta \vdash s_1 = s_2$ or $\Delta \vdash s_2 < s_1$).

Calculus *TFLTL* can be used to check the validity of formula. Formula F is valid iff it is possible to construct a derivation tree in calculus *TFLTL* of formula $\neg F$, in which every branch is closed. However S. Cerrito and M. Cialdea in 1997 described how *TFLTL* can be used to find a model of formula. A branch of derivation search tree of formula F is a model of formula F iff it is open and expanded.

Example 5.3.7. Let's find a model of formula $\Box\Diamond p$:



One branch is closed, because $s_0 \leqslant s_1$ contradicts to $s_1 < s_0$ and thus Δ is contradictory. However two other branches are open and expanded. This results in two models. The middle branch suggests a model with one state in which p is true. And the second branch gives a model with two states $s_0 < s_1$ such that p is true in s_1. The value of p in s_0 can be either true or false.

5.3.3 Planning problem

Let's describe how finite linear temporal logic can be used to solve planning problems. Suppose that the data of the problem and the goal are described using formulas of propositional logic. Next, the actions, that can be performed if certain conditions are met, are presented. Conditions are also defined using formulas of propositional logic.

The solution must provide a finite sequence of actions, that must be performed (and the order of the actions) to achieve the goal. If it is impossible to solve the problem (i.e. to achieve the goal using defined actions with given initial conditions), then the result must be a failure.

To define the problem, formulas of temporal logic with operators $\Box$, $\Diamond$ and $\circ$ are used.

The *initial condition* is described using some formula of propositional logic.

The *goal* is defined using formula $\Diamond G$, where G is formula of propositional logic. The meaning is *some time in the future G will hold.*

The *actions* are defined using formulas of the form $\Box\big(G \supset \mathrm{do}(a)\big)$, where G is a formula of propositional logic and a is an action, which can be performed. The meaning is *any time, if condition G holds, action a can be performed.*

In the result of performing an action, some *changes* in the data occur. They are defined using formula of the form $\Box\big(\mathrm{do}(a) \supset \circ G\big)$. Here G is a formula of propositional logic and a is an action, which was performed. The meaning is *any time, when action a have been performed, condition G holds.*

Restrictions are presented as formulas of the form $\Box G$. The meaning is *any time G holds.*

In some cases *incompatibility axioms* must also be included. They are of the following form:

$$\Box\Big(\,\mathrm{do}(a) \supset \big(\neg\,\mathrm{do}(a_1) \wedge \neg\,\mathrm{do}(a_2) \wedge \ldots \wedge \neg\,\mathrm{do}(a_n)\big)\Big)$$

The *result* is a sequence $\mathrm{do}(a_1), \mathrm{do}(a_2), \ldots, \mathrm{do}(a_m)$ or failure. If the solution was found, then it is clear that it is possible to reach the goal in m actions.

Chapter 6

Hybrid logic

6.1 Introduction

In 1990 P. Blackburn introduced a concept of *nominal* — the name of the node. The set of nominals $\mathcal{N} = \{i, j, \dots\}$ differs from the set of propositional variables $\mathcal{P} = \{p, q, \dots\}$. The same nominal can not denote several nodes. Nominals are also considered as atomic formulas, therefore expression $\Big(\Box_1(i \wedge p) \supset \Box_2(i \wedge q)\Big) \supset \Diamond_1(\neg p \vee \neg q)$ is also a hybrid formula. As it can be seen from this formula, nominals are usually incorporated into multimodal logic. For that purpose in most cases logic K_n is used.

Statement *in node i formula F is true* is denoted $@_i F$. Operator @ is called *satisfaction operator*. It satisfies the following conditions:

- distributivity: $@_i(F \supset G) \supset (@_i F \supset @_i G)$;
- necessity: if $\vdash F$, then $\vdash @_i F$;
- autoduality: $@_i F \equiv \neg @_i \neg F$.

Hybrid logic can be used to analyse structured data. Let's explain this with an example:

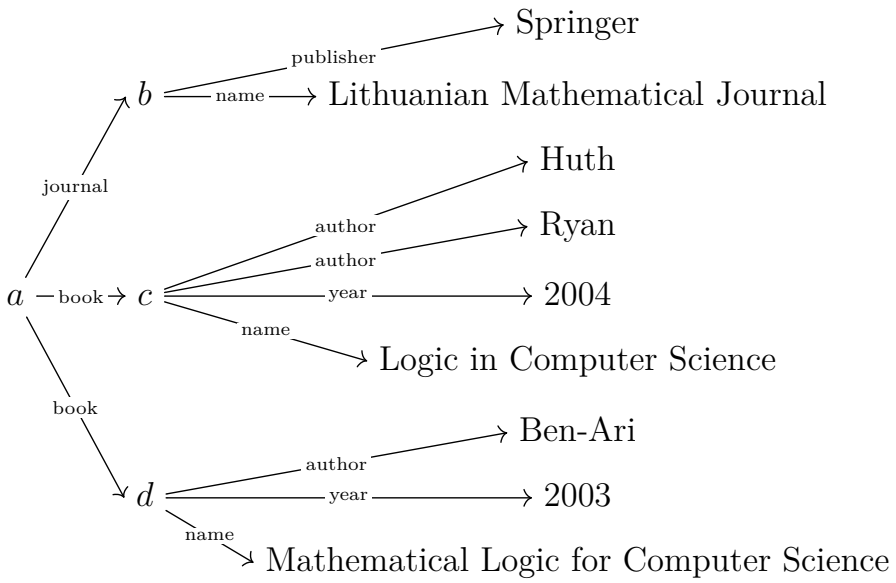
Example 6.1.1. Suppose a database of library is presented in an XML document. Only a small extract is presented here:

```
<journal>
   <publisher>Springer</publisher>
   <name>Lithuanian Mathematical Journal</name>
</journal>

<book>
   <author>Huth</author>
   <author>Ryan</author>
   <year>2004</year>
   <name>Logic in Computer Science</name>
</book>
```

```
<book>
   <author>Ben-Ari</author>
   <year>2003</year>
   <name>Mathematical Logic for Computer Science</name>
</book>
```

This XML can be presented as a tree:



For analysis of this data formulas of hybrid logic can be used. E.g. $\Box_{\text{author}}\neg$Kanger is true in node b, because Kanger is not among the authors of the book, defined in the node. Formula $\Box_{\text{journal}}\Box_{\text{name}}$LMJ (LMJ stands for *Lithuanian Mathematical Journal*) is true in node a. In node c formula $\Diamond_{\text{year}}\top$ is true, because there is an edge from c called *year*. For the same reason, formula $@_a\Box_{\text{book}}(\Diamond_{\text{author}}\top \vee \Diamond_{\textit{publisher}}\top)$ is true. Statement every book has at least one author can be defined with formula $@_a\Box_{\text{book}}\Diamond_{\text{author}}\top$.

6.2 Hybrid logic $\mathcal{H}(@,\downarrow)$

It was already mentioned that formulas of hybrid logics may contain propositional variables as well as nominals. Hybrid logic $\mathcal{H}(@,\downarrow)$ defines another operator $\downarrow$, called *bounding operator*. This operator marks the current state by some variable, which can be used later in the formula to denote the marked world. Thus another set of symbols is needed — set of nominal variables $\mathcal{X} = \{x, y, z, \dots\}$.

Definition 6.2.1. A *Formula* of hybrid logic $\mathcal{H}(@,\downarrow)$ is defined as follows:

- a propositional variable, nominal or nominal variable is a formula; it is also called *atomic formula*;
- if F is a formula, then $\neg F$, $\Box_k F$ and $\Diamond_k F$ are also formulas;

- if F and G are formulas, then $(F \wedge G)$, $(F \vee G)$, $(F \supset G)$ and $(F \leftrightarrow G)$ are also formulas;
- if i is a nominal and F is a formula, then $@_i F$ is also a formula;
- if x is a nominal variable and F is a formula, then $@_x F$ and $\downarrow x.F$ are also formulas.

A Kripke structure for hybrid logic is defined in a similar way as the one for multimodal logic in Definition 4.8.3.

Definition 6.2.2. A *Kripke structure* for hybrid logic formula F is a multiple $\langle \mathcal{W}, \mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_n, \nu \rangle$, where:

- $\mathcal{W}$ is a non empty set of worlds (states);
- $\mathcal{R}_i, i \in [1, n]$ is a binary relation in set $\mathcal{W}$ for modalities $\Box_i$ and $\Diamond_i$;
- $\nu : (\mathcal{P} \cup \mathcal{N}) \times \mathcal{W} \to \{\top, \bot\}$ is an interpretation function such that every nominal is true in one world of $\mathcal{W}$ only (i.e. suppose $i \in \mathcal{N}$ is a nominal, $w \in \mathcal{W}$ is some world and $\nu(i, w) = \top$, then for every other $w' \in \mathcal{W}, w' \neq w$ it is true that $\nu(i, w') = \bot$).

Additionally, to define, when hybrid formula is true, an *assignment function* is used. The fact that formula F is true in world w of structure $\mathcal{S}$ with assignment function g is denoted $\mathcal{S}, g, w \vDash F$. An assignment function (or simply *assignment*) $g : \mathcal{X} \to \mathcal{W}$ is a function that assigns a world to every nominal variable. A *variant* g^x_w of assignment g, where $x \in \mathcal{X}$ and $w \in \mathcal{W}$, is defined as follows:

$$g^x_w(y) = \begin{cases} w & \text{if } y = x \\ g(y) & \text{if } y \neq x \end{cases}$$

Definition 6.2.3. Suppose $\mathcal{S} = \langle \mathcal{W}, \mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_n, \nu \rangle$ is a Kripke structure for formula F, $w \in \mathcal{W}$ is some world and g is some assignment function. Now $\mathcal{S}, g, w \vDash F$ is defined recursively:

- if F is a propositional variable or a nominal (i.e. $F \in \mathcal{P}$ or $F \in \mathcal{N}$), then $\mathcal{S}, g, w \vDash F$ iff $\nu(F, w) = \top$;
- if F is a nominal variable (i.e. $F \in \mathcal{X}$), then $\mathcal{S}, g, w \vDash F$ iff $g(x) = w$;
- if $F = \neg G$, then $\mathcal{S}, g, w \vDash F$ iff $\mathcal{S}, g, w \nvDash G$;
- if $F = G \wedge H$, then $\mathcal{S}, g, w \vDash F$ iff $\mathcal{S}, g, w \vDash G$ and $\mathcal{S}, g, w \vDash H$;
- if $F = G \vee H$, then $\mathcal{S}, g, w \nvDash F$ iff $\mathcal{S}, g, w \nvDash G$ and $\mathcal{S}, g, w \nvDash H$;
- if $F = G \supset H$, then $\mathcal{S}, g, w \nvDash F$ iff $\mathcal{S}, g, w \vDash G$ and $\mathcal{S}, g, w \nvDash H$;
- if $F = G \leftrightarrow H$, then $\mathcal{S}, g, w \vDash F$ iff $\mathcal{S}, g, w \vDash G$ and $\mathcal{S}, g, w \vDash H$ or $\mathcal{S}, g, w \nvDash G$ and $\mathcal{S}, g, w \nvDash H$;
- if $F = \Box_l G, l \in [1, n]$, then $\mathcal{S}, g, w \vDash F$ iff for every $w' \in \mathcal{W}$ such that $w \mathcal{R}_l w'$ it is true that $\mathcal{S}, g, w' \vDash G$;

- if $F = \Diamond_l G, l \in [1, n]$, then $\mathcal{S}, g, w \vDash F$ iff there is $w' \in \mathcal{W}$ such that $w \mathcal{R}_l w'$ and $\mathcal{S}, g, w' \vDash G$;
- if $F = @_i G, i \in \mathcal{N}$ and $\nu(i, w') = \top$, then $\mathcal{S}, g, w \vDash F$ iff $\mathcal{S}, g, w' \vDash G$;
- if $F = @_x G$ where $x \in \mathcal{X}$, then $\mathcal{S}, g, w \vDash F$ iff $\mathcal{S}, g, g(x) \vDash G$;
- if $F = \downarrow x.G$, then $\mathcal{S}, g, w \vDash F$ iff $\mathcal{S}, g^x_w, w \vDash G$;

Example 6.2.4. Let's define several properties of relations using hybrid logic formulas:

- Irreflexivity: $@_x \neg \Diamond_l x$ or $\downarrow x.\Box_l \neg x$;
- Asymmetry: $@_x \neg \Diamond_l \Diamond_l x$ or $\downarrow x.\Box_l \Box_l \neg x$;
- Antisymmetry: $@_x \Box_l (\Diamond_l x \supset x)$ or $\downarrow x.\Box_l (\Diamond_l x \supset x)$;
- Transitivity: $@_x (\Diamond_l \Diamond_l x \supset \Diamond_l x)$ or $\downarrow x.\Box_l \Box_l \downarrow y.@_x \Diamond_l y$;
- Density[1]: $@_x \Diamond_l x \supset \Diamond_l \Diamond_l x$ or $\downarrow x.\Box_l \downarrow y.@_x \Diamond_l \Diamond_l y$;
- Trichotomy[2]: $@_x \Diamond_l y \vee @_x y \vee @_y \Diamond_l x$

6.3 Relation to predicate logic

Formulas of hybrid logic $\mathcal{H}(@, \downarrow)$ can be transformed in to predicate logic in the similar manner as formulas of modal (or multimodal logic). That is, for every formula F of $\mathcal{H}(@, \downarrow)$ it is possible to find formula $\mathrm{Tr}(F)_x$ with one free variable x such that F is satisfiable in $\mathcal{H}(@, \downarrow)$ iff $\mathrm{Tr}(F)_x$ is satisfiable in predicate logic with equality predicate. Let's describe the algorithm:

- $\mathrm{Tr}(F)_x = (x = F)$, if $F \in \mathcal{N}$ or $F \in \mathcal{X}$;
- $\mathrm{Tr}(F)_x = P(x)$, if $F \in \mathcal{P}$, here P is predicate variable that corresponds to propositional variable F;
- $\mathrm{Tr}(\neg G)_x = \neg \mathrm{Tr}(G)_x$;
- $\mathrm{Tr}(G \wedge H)_x = \mathrm{Tr}(G)_x \wedge \mathrm{Tr}(H)_x$;
- $\mathrm{Tr}(G \vee H)_x = \mathrm{Tr}(G)_x \vee \mathrm{Tr}(H)_x$;
- $\mathrm{Tr}(G \supset H)_x = \mathrm{Tr}(G)_x \supset \mathrm{Tr}(H)_x$;
- $\mathrm{Tr}(\Box_k G)_x = \forall z \Big(R_k(x, z) \supset \mathrm{Tr}(G)_z \Big)$;
- $\mathrm{Tr}(\Diamond_k G)_x = \exists z \Big(R_k(x, z) \wedge \mathrm{Tr}(G)_z \Big)$;
- $\mathrm{Tr}(@_y G)_x = \mathrm{Tr}(G)_y$, if $y \in \mathcal{N}$ or $y \in \mathcal{X}$;
- $\mathrm{Tr}(\downarrow y.G)_x = \mathrm{Tr}(G)_x \{^x/_y\}$, here $\{^x/_y\}$ is a substitution.

[1] $\forall x \forall y \Big(\mathcal{R}_k(x, y) \supset \exists z \big(\mathcal{R}_k(x, z) \wedge \mathcal{R}_k(z, y) \big) \Big)$

[2] $\forall x \forall y \big(\mathcal{R}_k(x, y) \vee (x = y) \vee \mathcal{R}_k(y, x) \big)$

Here z is a new variable.

Example 6.3.1. Let's transform formula $\downarrow x.\Box_k\Box_k\downarrow y.@_x\Diamond_k y$ (transitivity) into predicate logic:

$$\mathrm{Tr}(\downarrow x.\Box_k\Box_k\downarrow y.@_x\Diamond_k y)_{x'} = \mathrm{Tr}(\Box_k\Box_k\downarrow y.@_x\Diamond_k y)_{x'}\{x'/x\} =$$

$$= \forall y'\big(\mathcal{R}_k(x',y') \supset \mathrm{Tr}(\Box_k\downarrow y.@_x\Diamond_k y)_{y'}\big)\{x'/x\} =$$

$$= \forall y'\Big(\mathcal{R}_k(x',y') \supset \forall z'\big(\mathcal{R}_k(y',z') \supset \mathrm{Tr}(\downarrow y.@_x\Diamond_k y)_{z'}\big)\Big)\{x'/x\} =$$

$$= \forall y'\Big(\mathcal{R}_k(x',y') \supset \forall z'\big(\mathcal{R}_k(y',z') \supset \mathrm{Tr}(@_x\Diamond_k y)_{z'}\{z'/y\}\big)\Big)\{x'/x\} =$$

$$= \forall y'\Big(\mathcal{R}_k(x',y') \supset \forall z'\big(\mathcal{R}_k(y',z') \supset \mathrm{Tr}(\Diamond_k y)_x\{z'/y\}\big)\Big)\{x'/x\} =$$

$$= \forall y'\Big(\mathcal{R}_k(x',y') \supset \forall z'\Big(\mathcal{R}_k(y',z') \supset \exists u\big(\mathcal{R}_k(x,u) \wedge \mathrm{Tr}(y)_u\big)\{z'/y\}\Big)\Big)\{x'/x\} =$$

$$= \forall y'\Big(\mathcal{R}_k(x',y') \supset \forall z'\Big(\mathcal{R}_k(y',z') \supset \exists u\big(\mathcal{R}_k(x,u) \wedge u = y\big)\{z'/y\}\Big)\Big)\{x'/x\} =$$

$$= \forall y'\Big(\mathcal{R}_k(x',y') \supset \forall z'\Big(\mathcal{R}_k(y',z') \supset \exists u\big(\mathcal{R}_k(x,u) \wedge u = z'\big)\Big)\Big)\{x'/x\} =$$

$$= \forall y'\Big(\mathcal{R}_k(x',y') \supset \forall z'\Big(\mathcal{R}_k(y',z') \supset \exists u\big(\mathcal{R}_k(x',u) \wedge u = z'\big)\Big)\Big)$$

Example 6.3.2. Let's transform formula $\downarrow x.\Box_k(\Diamond_k x \supset x)$ (antisymmetry) into predicate logic:

$$\mathrm{Tr}\left(\downarrow x.\Box_k(\Diamond_k x \supset x)\right)_{x'} = \mathrm{Tr}\left(\Box_k(\Diamond_k x \supset x)\right)_{x'}\{x'/x\} =$$

$$= \forall y\big(\mathcal{R}_k(x',y) \supset \mathrm{Tr}(\Diamond_k x \supset x)_y\big)\{x'/x\} =$$

$$= \forall y\Big(\mathcal{R}_k(x',y) \supset \big(\,\mathrm{Tr}(\Diamond_k x)_y \supset \mathrm{Tr}(x)_y\big)\Big)\{x'/x\} =$$

$$= \forall y\Big(\mathcal{R}_k(x',y) \supset \Big(\exists z\big(\mathcal{R}_k(y,z) \wedge \mathrm{Tr}(x)_z\big) \supset y = x\Big)\Big)\{x'/x\} =$$

$$= \forall y\Big(\mathcal{R}_k(x',y) \supset \Big(\exists z\big(\mathcal{R}_k(y,z) \wedge z = x\big) \supset y = x\Big)\Big)\{x'/x\} =$$

$$= \forall y\Big(\mathcal{R}_k(x',y) \supset \Big(\exists z\big(\mathcal{R}_k(y,z) \wedge z = x'\big) \supset y = x'\Big)\Big)$$

Example 6.3.3. Let's transform formula $\downarrow x.\Box_k\downarrow y.@_x\Diamond_k\Diamond_k y$ (density) into predicate logic:

$$\mathrm{Tr}(\downarrow x.\Box_k\downarrow y.@_x\Diamond_k\Diamond_k y)_{x'} = \mathrm{Tr}(\Box_k\downarrow y.@_x\Diamond_k\Diamond_k y)_{x'}\{x'/x\} =$$

$$= \forall y'\big(\mathcal{R}_k(x',y') \supset \mathrm{Tr}(\downarrow y.@_x\Diamond_k\Diamond_k y)_{y'}\big)\{x'/x\} =$$

$$= \forall y'\big(\mathcal{R}_k(x',y') \supset \mathrm{Tr}(@_x\Diamond_k\Diamond_k y)_{y'}\{y'/y\}\big)\{x'/x\} =$$

$$= \forall y'\big(\mathcal{R}_k(x',y') \supset \mathrm{Tr}(\Diamond_k\Diamond_k y)_x\{y'/y\}\big)\{x'/x\} =$$

$$= \forall y'\Big(\mathcal{R}_k(x',y') \supset \exists z\big(\mathcal{R}_k(x,z) \wedge \mathrm{Tr}(\Diamond_k y)_z\big)\{y'/y\}\Big)\{x'/x\} =$$

$$= \forall y'\Big(\mathcal{R}_k(x',y') \supset \exists z\Big(\mathcal{R}_k(x,z) \wedge \exists u\big(\mathcal{R}_k(z,u) \wedge \mathrm{Tr}(y)_u\big)\Big)\{y'/y\}\Big)\{x'/x\} =$$

$$= \forall y'\Big(\mathcal{R}_k(x',y') \supset \exists z\Big(\mathcal{R}_k(x,z) \wedge \exists u\big(\mathcal{R}_k(z,u) \wedge u = y\big)\Big)\{y'/y\}\Big)\{x'/x\} =$$

$$= \forall y'\Big(\mathcal{R}_k(x',y') \supset \exists z\Big(\mathcal{R}_k(x,z) \wedge \exists u\big(\mathcal{R}_k(z,u) \wedge u = y'\big)\Big)\Big)\{x'/x\} =$$

$$= \forall y'\Big(\mathcal{R}_k(x',y') \supset \exists z\Big(\mathcal{R}_k(x',z) \wedge \exists u\big(\mathcal{R}_k(z,u) \wedge u = y'\big)\Big)\Big)$$

Let's analyse another hybrid logic — $\mathcal{H}(@,\forall)$. In this logic operator $\forall$ is included instead of operator $\downarrow$. Formula $\forall xF$ is true, if formula F is true in every world x of the Kripke structure. In 2000 C. Areces proved that logic $\mathcal{H}(@,\forall)$ is equivalent to predicate logic with equality predicate. It follows from the fact that every predicate logic formula, which contains only one-place and two-place predicate variables can be transformed into $\mathcal{H}(@,\forall)$. Let the set of one place predicate variables of the formula be $P_1, P_2, \ldots$ and the set of two-place predicate variables be $R_1, R_2, \ldots$. Let's describe the transformation:

- $\mathrm{Tr}\big(P_k(x)\big) = @_x p_k$, where p is propositional variable, which represents predicate variable P;
- $\mathrm{Tr}\big(R_k(x_1,x_2)\big) = @_{x_1}\Diamond_k x_2$;
- $\mathrm{Tr}(x_1 = x_2) = @_{x_1}x_2$;
- $\mathrm{Tr}(\neg F) = \neg\,\mathrm{Tr}(F)$;
- $\mathrm{Tr}(F \wedge G) = \mathrm{Tr}(F) \wedge \mathrm{Tr}(G)$;
- $\mathrm{Tr}(F \vee G) = \mathrm{Tr}(F) \vee \mathrm{Tr}(G)$;
- $\mathrm{Tr}(F \supset G) = \mathrm{Tr}(F) \supset \mathrm{Tr}(G)$;
- $\mathrm{Tr}(\forall xF) = \forall x\,\mathrm{Tr}(F)$;
- $\mathrm{Tr}(\exists xF) = \neg\forall x\neg\,\mathrm{Tr}(F)$;

Predicate formula F is satisfiable in predicate logic iff formula $\mathrm{Tr}(F)$ is satisfiable in hybrid logic $\mathcal{H}(@, \forall)$.

The analogous transformation or predicate formulas into hybrid logic $\mathcal{H}(@, \downarrow)$ is not known, however such transformation is possible for a group of predicate logic formulas. The analysed formulas may contain individual constants, equality predicate, one two-place predicate R and any number of one-place predicates $P_1, P_2, \ldots$.

Definition 6.3.4. A *limited formula* is defined as follows:

- if t is a term (a term is an individual constant or individual variable), then $P_i(t), i \in [1, \infty)$ is a limited formula;
- if t_1 and t_2 are terms, then $R(t_1, t_2)$ and $t_1 = t_2$ are limited formulas;
- if F is a limited formula, then $\neg F$ is also a limited formula;
- if F and G are limited formulas, then $(F \wedge G)$ is also a limited formula;
- if F is a limited formula, t is a term and x is an individual variable such that $x \neq t$, then $\exists x\Big(R(t, x) \wedge F\Big)$ is also a limited formula.[3]

In 1999 C. Areces, P. Blackburn and M. Marx described the transformation of limited formulas into hybrid logic $\mathcal{H}(@, \downarrow)$:

- $\mathrm{Tr}\Big(R(t_1, t_2)\Big) = @_{t_1} \Diamond t_2$;
- $\mathrm{Tr}\Big(P_i(t)\Big) = @_t p_i$;
- $\mathrm{Tr}(t_1 = t_2) = @_{t_1} t_2$;
- $\mathrm{Tr}(\neg F) = \neg\, \mathrm{Tr}(F)$;
- $\mathrm{Tr}(F \wedge G) = \mathrm{Tr}(F) \wedge \mathrm{Tr}(G)$;
- $\mathrm{Tr}\Big(\exists x\Big(R(t, x) \wedge F\Big)\Big) = @_t \Diamond \downarrow x.\, \mathrm{Tr}(F)$.

Here t_1 and t_2 are terms, x is a variable.

It was proved that any limited formula F is satisfiable in predicate logic iff $\mathrm{Tr}(F)$ is satisfiable in $\mathcal{H}(@, \downarrow)$.

[3] Note, that $\exists x\Big(R(x, x) \wedge (x = x)\Big)$ is not a limited formula.

Bibliography

[1] R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning About Knowledge*. The MIT Press, 2003.

[2] R. Feys. *Modal Logics*. E. Nauwelaerts, Louvain, 1965.

[3] J. Gaintzarain, M. Hermo, P. Lucio, M. Navarro, and F. Orejas. A cut-free and invariant-free sequent calculus for PLTL. In J. Duparc and T. A. Henzinger, (Eds.), *Computer Science Logic*, volume 4646 of *Lecture Notes in Computer Science*, pp. 481–495. Springer, 2007.

[4] G. Gentzen. Untersuchungen über das logische Schließen. I. *Mathematische Zeitschrift*, **39**, pp. 176–221, 1935.

[5] G. Gentzen. Investigations into logical deduction. *American Philosophical Quarterly*, **1**(4), pp. 288–306, 1964. Originally published in [4]. English translation by M.E. Szabo.

[6] R. Goré. Chapter 6: Tableau methods for modal and temporal logics. In M. D'Agostino, D. M. Gabbay, R. Hähnle, and J. Posegga, (Eds.), *Handbook of Tableau Methods*, pp. 297–396. Kluwer Academic Publishers, Dordrecht, 1999.

[7] D. Hilbert. Die Grundlagen der Mathematik. *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg*, **6**(1), pp. 65–85, 1928.

[8] G. E. Hughes and M. J. Cresswell. *An Introduction to Modal Logic*. Methuen and Co, London, 1985.

[9] S. Kanger. *Provability in logic*. Almgvist & Wiksell, Stockholm, 1957.

[10] S. C. Kleene. *Introduction to Metamathematics*. D. van Nostrand Company Inc., Princeton, New Jersey, 1950.

[11] J.-J. C. Meyer and W. van der Hoek. *Epistemic Logic for AI and Computer Science*, volume 41 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 1995.

[12] G. F. Shvarts. Gentzen style systems for K45 and K45D. In A. R. Meyer and M. A. Taitslin, (Eds.), *Logic at Botic '89*, volume 363 of *Lecture Notes in Computer Science*, pp. 245–256. Springer-Verlag, Pereslavl-Zalessky, 1989.

[13] A. S. Troelstra and H. Schwichtenberg. *Basic Proof Theory.* Cambridge University Press, 1996.