Technical University of Denmark

Course 01017: Discrete Mathematics

Lecture notes on Classical Logic

Valentin Goranko

Fall 2011

Introduction

Mathematical logic is a modern, vigorously developing and expanding branch of mathematics, which studies

- the logical foundations, laws and structure of mathematical reasoning and proofs,
- logical reasoning in general, using mathematical methods,
- the applications of logical methods and results to mathematics.

Contemporary logic has fundamental applications not only to mathematics, but also to a number of other modern disciplines, including:

- *Computer science,* to which it gives theoretical foundations and formal methods for analysis of basic concepts such as *computation*, *algorithms and their complexity, semantics of programming languages*, as well as practical tools for software and hardware synthesis and verification, database management, logic programming, etc.

  The impact of logic on computer science is comparable to the impact of calculus on physics and engineering.
- *Artificial intelligence,* where it provides methodology and tools for automated reasoning of intelligent agents.
- *Linguistics*, where logical methods are instrumental for analysis of natural and artificial languages.

The purpose of these notes is to give you a sound introductory background to the basic concepts of classical logic to an extent that would enable you to properly express mathematical statements in a formal logical language, to analyze their logical form, meaning, and truth, and to perform correct mathematical reasoning using logic.

We distinguish two levels of logical expressions and reasoning:

- The lower level is *propositional logic.* Propositions are sentences built, using *propositional logical connectives,* from primitive statements with no internal logical structure but only a *truth value* (`true` or `false`). Propositional logic can formalize fragments of common reasoning, but it is certainly insufficient for mathematical reasoning.
- The higher level is *predicate* (or *first-order) logic* which builds on propositional logic by using *constants (names)* and *variables for objects* (numbers, sets, human beings, etc.), *functions and predicates* over objects, as well as *quantifiers* over them, like: 'for all objects x (. . .x . . .)', and 'there exists an object x such that (. . . x . . .)'. First-order logic is, in a way, sufficient to express any mathematical statement. That is why it is usually regarded as *the* logic of mathematics.

In these notes we introduce and discuss the *formal languages* of propositional and first-order logic, the notions of *truth, validity, and logical consequence.* We also discuss the concept of a *deductive system*, meant to formalize and mechanize logical reasoning, and introduce and illustrate the use of the main types of deductive systems. At the end of these notes we illustrate the use of logic for mathematical reasoning and proofs.

Contents

1 Propositional Logic

1.1 Propositions and logical connectives. Truth-tables and tautologies.

1.1.1 Propositions

The basic concept of propositional logic is **proposition**. A proposition is a sentence which can be assigned a **truth value** : `true` or `false`.

Some simple examples:

- The Sun is hot.
- The Earth is made of cheese.
- 2 plus 2 equals 22.
- The 2011-th decimal digit of the number π is 9.

And here are some sentences which are not propositions (why?):

- Are you bored?
- Please, don't go away!
- She loves him.
- x is an integer.
- This sentence is false.

1.1.2 Propositional logical connectives

The propositions above have no internal logical structure and will be called *primitive*, or *atomic* propositions. From primitive propositions one can form *compound* ones by using *logical connectives*. The most commonly used connectives are:

- **not**, called *negation*, denoted by $\neg$;
- **and**, called *conjunction*, denoted by $\wedge$ (or sometimes, by &);
- **or**, called *disjunction*, denoted by $\vee$;
- **if** ... **then** ... , called *implication*, or *conditional*, denoted by $\rightarrow$;
- **. . . if and only if . . .** , called *biconditional* , denoted by $\leftrightarrow$;

Remark 1 *It is usually not grammatically correct to read compound propositions by simply* *inserting the names of the logical connectives in between the atomic components. A typical problem arises with the negation: one does not say "Not the earth is square". A uniform way to get round that difficulty and to negate a proposition P is to say "It is not the case that P".*

Thus, given the propositions

"Two plus two equals five." and "The Sun is hot."

we can form the propositions

- "It is **not** the case that two plus two equals five. "
- "Two plus two equals five **and** the Sun is hot."
- "Two plus two equals five **or** the Sun is hot."
- "**If** two plus two equals five **then** the Sun is hot."
- "Two plus two equals five **if and only if** the Sun is hot."

For a more involved example, from the propositions:

"Mary is clever.", "Mary is lazy.", and "Mary likes logic."

we can compose a proposition (smoothed out a bit) like:

"Mary is not clever or if she likes logic then she is clever and not lazy."

1.1.3 Truth-tables

How about the truth-value of a compound proposition? It can be *calculated* from the truth-values of the components (much in the same way as we can calculate the value of the algebraic expression $A \times (B - C) + B/A$ as soon as we know the values of A, B, C) by following the rules of 'propositional arithmetic':

- *The proposition $\neg A$ is true if and only if*

  the proposition A is false.
- *The proposition $A \wedge B$ is true if and only if*

  both A and B are true.
- *The proposition $A \vee B$ is true if and only if*

  either of A or B (possibly both) is true.
- *The proposition $A \rightarrow B$ is true if and only if*

  A is false or B is true, i.e. if the truth of A implies the truth of B.
- *The proposition $A \leftrightarrow B$ is true if and only if*

  A and B have the same truth-values.

We can systematize these rules in sort of "multiplication tables". For that purpose, and to make it easier for symbolic (that is, mathematical) manipulations, we introduce a special notation for the two truth values by denoting the value '`true`' by `T`, and the value ' `false`' by `F`. Another common notation, particularly in computer science, is to denote `true` by **1** and `false` by **0**.

Now the rules of the "propositional arithmetic" can be tabulated by means of the following *truth-tables* (p and q below stand for arbitrary propositions):

p	$\neg p$
T	F
F	T

p	q	$p \wedge q$	$p \vee q$	$p \to q$	$p \leftrightarrow q$
T	T	T	T	T	T
T	F	F	T	F	F
F	T	F	T	T	F
F	F	F	F	T	T

1.1.4 Some remarks on the meaning of the connectives

The use and meaning of the logical connectives in natural language does not always match their formal logical meaning. For instance, quite often the conjunction is loaded with a causal relationship which makes the meanings of the sentences "He threw the stone and the window broke." and " The window broke and he threw the stone." quite different, while they have the same truth value (see the truth table of the conjunction). Further, the disjunction is often used in an *exclusive* sense, e.g. in "I shall win or I shall die.", while in formal logic we use it by convention in an *inclusive* sense, and thus "You will win or I will win." will be true if we both win.

Amongst all logical connectives, however, the implication seems to be the most questionable one. For instance, it is not so easy to accept that a proposition such as "If 2+2=5, then the moon is made of cheese.", if it makes any sense at all, should be true. The leading motivation in determining the truth behavior of the implication is, of course, the logical meaning we assign to it. The proposition $A \to B$ means:

***If** A is true, **then** B must be true.*

NB: Note that if A is not true, then the (truth of the) implication $A \to B$ requires *nothing* regarding the truth of B. One can think of an implication as a promise, e.g., Johnnie's father tells him: "**If** you pass your logic exam, **then** I'll buy you a motorbike." Now consider the four possible situations (Johnnie passes/fails his exam and his father buys/does not buy him a motorbike) and see in which of them the promise is kept (the implication is true) and in which it is broken (the implication is false). That argument should lead you to the truth-table of the implication. It is very important to understand and remember that truth-table, since the implication is the logical connective which is most closely related to the concepts of logical reasoning and deduction.

Some terminology: the proposition A in the implication $A \to B$ is called the *antecedent* and the proposition B is the *consequent* of the implication.

The implication $A \to B$ can be expressed in many different but logically equivalent ways, which you should be able to recognize:

- A implies B.
- B follows from A.
- If A, (then) B.
- A only if B.
- B if A.
- B whenever A.
- A is sufficient for B.

  (*Meaning: The truth of A is sufficient for the truth of B.*
- B is necessary for A.

  (*Meaning: The truth of B is necessary for A to be true.*

1.1.5 Computing truth-values of propositions

You can see from the truth-tables that *the truth-value of a compound proposition does not depend on the meaning of the component propositions, but only on their truth-values.* So, to check the truth of such a proposition we merely need to replace all component propositions by their respective truth-values, and then 'calculate' the truth of the whole proposition, using the truth tables of the logical connectives. Thus,

- "It is not the case that two plus two equals five." is true;
- "Two plus two equals five and the Sun is hot." is false;
- "Two plus two equals five or the Sun is hot." is true;
- "If two plus two equals five, then the Sun is hot." is true (even though it does not make much sense).

For the last example, suppose we know that

> "Mary is clever." is true,
> "Mary is lazy." is false,
> "Mary likes logic." is true.

Then the truth-value of the compound proposition

> "Mary is not clever or if she likes logic, then she is clever and not lazy."

can be determined just as easily. However, in order to do so, we first have to analyze the *syntactic structure* of the proposition, i.e. to determine how it has been composed, that is, in what order the logical connectives occurring therein have been applied. With algebraic expressions such as $A \times (B - C) + B/A$ that analysis is a little easier thanks to the use of parentheses and the established priority order amongst the arithmetic operations. Let us, too, make use of parentheses and rewrite the sentence in the way we (presumably) all understand it:

"(Mary is not clever) or (if (she likes logic), then ((she is clever) and (not lazy)))."

(We could, by the same token, impose some priority order amongst the logical connectives, but we won't bother now.) The structure of the sentence should be clear now. We can, however, go one step further and make it look exactly like an algebraic expression by using letters to denote the occurring primitive propositions. Let us, for instance, denote

"Mary is clever." by A,
"Mary is lazy." by B,
"Mary likes logic." by C.

Now our compound proposition can be neatly rewritten as

$$(\neg A) \vee (C \to (A \wedge \neg B)).$$

In our rather informal exposition we shall not use parentheses very systematically, but only whenever necessary to avoid ambiguity. For that purpose we will, as in arithmetic, impose a priority order amongst the logical connectives, viz.:

- the negation has the strongest binding power, i.e. the highest priority,
- then come the conjunction and disjunction,
- then the implication, and
- the biconditional has the lowest priority.

Example 2 *The proposition* $\neg A \vee C \to A \wedge \neg B$ *is an abbreviation of* $((\neg A) \vee C) \to (A \wedge (\neg B))$.

And now, the last step is to calculate the truth-value. Here we make use of our observation above and simply replace the atomic propositions A, B, and C by their truth-values and do the formal calculation following the truth-tables step-by-step:

$$(\neg \mathtt{T}) \vee (\mathtt{T} \to (\mathtt{T} \wedge \neg \mathtt{F})) = \mathtt{F} \vee (\mathtt{T} \to (\mathtt{T} \wedge \ \mathtt{T})) = \mathtt{F} \vee (\mathtt{T} \to \mathtt{T}) = \ \mathtt{F} \vee \mathtt{T} = \mathtt{T}.$$

1.1.6 Propositional formulae and their truth-tables

If we only discuss particular propositions, our study of logic would be no more useful than a study of algebra based on particular equalities such as $2+3 = 5$ or $12345679 \times 9 = 111111111$. Instead, we should look at *schemes of propositions* and their properties, just like we study algebraic formulae and equations and their properties. We call such schemes of propositions **propositional formulae**.

Propositional formulae: basics

To begin with, we will specify a *formal language*, in which propositional formulae, meant to be templates for composite propositions, will be special words. That language involves:

- **propositional constants**: special fixed propositions $\top$ (which always takes a truth-value `true`), and $\bot$ (which always takes a value `false`),

- **propositional variables** $p, q, r \ldots$, possibly indexed, to denote unspecified propositions, just in the same way as we use algebraic variables to denote unspecified or unknown numbers;
- the **logical connectives** that we already know,
- **auxiliary symbols**: parentheses '(' and ')'. We use these to indicate the order of application of logical connectives and make the formulae unambiguous.

Using these ingredient we can construct propositional formulae, the way in which we are used to constructing algebraic expressions from variables and arithmetic operations. Here are a few examples of propositional formulae:

1. Every propositional constant or variable is a propositional formula.
2. If A is a propositional formula then $\neg A$ is a propositional formula.
3. If A, B are propositional formulae then $(A \lor B)$, $(A \land B)$, $(A \to B)$, $(A \leftrightarrow B)$ are propositional formulae.

Hereafter we will be omitting the 'unnecessary' pairs of parentheses, whenever that would not lead to syntactic ambiguity.

Note that the notion of propositional formula is used in its own definition; that is the idea of *recursion.* The definition works as follows: the first rule above gives us some initial stock of propositional formulae; as we keep applying the other rules, we build more and more of them, and every propositional formula can be obtained in several (finitely many!) steps of applying these rules.

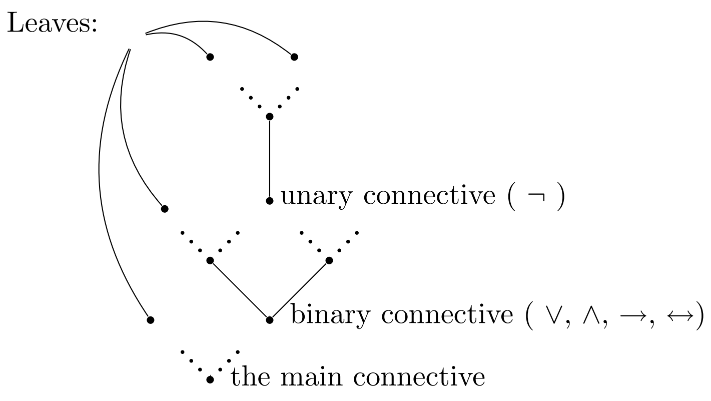
The formulae which occur in the process of the construction of a formula A are called **subformulae of** A. The last propositional connective introduced in the construction of A is called **the main connective of** A and the formula(e) to which is is applied is/are **the main subformula(e) of** A.

Construction and parsing tree of a formula

The sequence of formulae constructed in the process of applying the definition and ending with A is called a *construction sequence of a formula A.* A formula has many construction sequences, and a construction sequence may contain many redundant formulae. so, a better notion capturing the construction of a formula is the one of a *construction tree* of that formula. A construction tree is a tree-like (directed) graph with nodes labelled with propositional constants, variables, and propositional connectives, such that:

1. Every leaf is labelled by a propositional constant or variable.
2. Propositional constants and variables label only leaves.
3. Every node labelled with $\neg$ has exactly one successor node.
4. Every node labelled with any of $\land, \lor, \to, \leftrightarrow$ has exactly two successor nodes - *left* and *right* successor.

Thus, a construction tree looks like this:

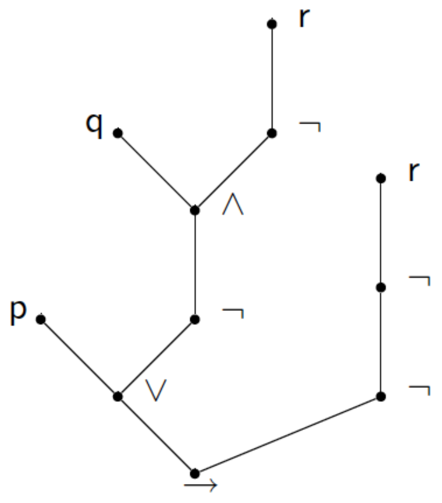


Every construction tree defines a formula C, built starting from the leaves and going towards the root, by applying at every node the formula construction rule corresponding to the label at that node. The formulae constructed in the process are precisely the subformulae of C and the propositional connective labeling the root of the construction tree of a formula C is the main connective of C.

Example 3 (Construction tree) *Formula:*

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

Construction tree:



The *parsing tree* of a formula looks the same as its construction tree, but is produced in inverse order, starting from the main connective (of any), drawing edges to the main components, and then recursively producing the parsing trees for each of them.

Truth assignments. Satisfaction of propositional formulae.

A propositional formula is a scheme, that becomes a proposition whenever we substitute propositions for all occurring propositional variables. Of course, we mean *uniform* substitutions, that is, the same variables are replaced by the same propositions throughout the formula.

We cannot attribute a truth-value to a propositional formula before we assign concrete propositions, or at least concrete *truth-values*, to all occurring propositional variables – for the same reason that we cannot evaluate $x(y+z)$ before we have assigned values to x, y, z.

For instance, if we substitute "0.5 is an integer" for p, " 2 is less than 3" for q and "The Moon is larger than the sun" for r in the propositional formula

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

we find that the resulting proposition is true:

$(\mathtt{F} \vee \neg(\mathtt{T} \wedge \neg\mathtt{F})) \to \neg\neg\mathtt{F} \;=\; (\mathtt{F} \vee \neg(\mathtt{T} \wedge \mathtt{T})) \to \neg\mathtt{T} \;=\; (\mathtt{F} \vee \neg\mathtt{T}) \to \mathtt{F}$

$=\; (\mathtt{F} \vee \mathtt{F}) \to\; \mathtt{F} \;=\; \mathtt{F} \to \mathtt{F} \;=\; \mathtt{T}.$

If, however, we substitute any true propositions for p and q and a false proposition for r, then the resulting proposition will be false:

$(\mathtt{T} \vee \neg(\mathtt{T} \wedge \neg\mathtt{F})) \to \neg\neg\mathtt{F} \;=\; (\mathtt{T} \vee \neg(\mathtt{T} \wedge \mathtt{T})) \to \neg\mathtt{T} \;=\; (\mathtt{T} \vee \neg\mathtt{T}) \to \mathtt{F}$

$=\; (\mathtt{T} \vee \mathtt{F}) \to\; \mathtt{F} \;=\; \mathtt{T} \to \mathtt{F} \;=\; \mathtt{F}$

Definition 4 *A combination of truth values assigned to a set propositional variables is called* ***truth assignment*** *for these variables. If a truth assignment τ renders a formula A true, we say that it τ* ***satisfies*** *A. To denote that we sometimes write $\tau \models A$. A propositional formula is* ***satisfiable*** *If it is satisfied by some truth assignment.*

For instance, the formula $p \wedge \neg q \wedge r$ is satisfiable by the assignment $p : \mathtt{T}$, $q : \mathtt{F}$, and $r : \mathtt{T}$, while the formula $p \wedge \neg p$ is not satisfiable.

Truth-tables of propositional formulae

Clearly, the truth of a given propositional formula only depends on the truth values assigned to the variables occurring in that formula. Thus, we can think of propositional formulae as functions from truth assignments to truth values and we can tabulate the 'behaviour' of any propositional formula in a truth-table where we list all possible truth assignments of the occurring variables, and for each of them we compute the corresponding truth-value of the formula. We can do that by successively computing the truth-values of all occurring subformulae, as we did just now. For example, the truth-table for the formula above is compiled as follows:

p	q	r	$\neg r$	$\neg\neg r$	$q \wedge \neg r$	$\neg(q \wedge \neg r)$	$p \vee \neg(q \wedge \neg r)$	$(p \vee \neg(q \wedge \neg r)) \to \neg\neg r$
T	T	T	F	T	F	T	T	T
T	T	F	T	F	T	F	T	F
T	F	T	F	T	F	T	T	T
T	F	F	T	F	F	T	T	F
F	T	T	F	T	F	T	T	T
F	T	F	T	F	T	F	F	T
F	F	T						
F	F	F						

The last two rows are left to the reader as exercises.

Truth-tables can be somewhat simplified if we notice that every occurrence of a logical connective in a formula determines a *unique subformula* where that occurrence is the **main connective** (i.e. the connective applied last in the construction of the subformula). For instance, the conjunction occurring in the formula $(p \vee \neg(q \wedge \neg r)) \to \neg\neg r$ is the main connective of the subformula $q \wedge \neg r$; the first negation is the main connective of $\neg(q \wedge \neg r)$, and the implication is the main connective of the whole formula. Now we can simplify the truth-table of the formula by listing only the occurring variables and the whole formula, and by computing the truth-table of every subformula in the column below the corresponding main connective:

p	q	r	$(p$	$\vee$	$\neg$	$(q$	$\wedge$	$\neg$	$r))$	$\to$	$\neg$	$\neg$	r
T	T	T	T	T	T	T	F	F	T	T	T	F	T
T	T	F	T	T	F	T	T	T	F	F	F	T	F
T	F	T	T	T	T	F	F	F	T	T	T	F	T
T	F	F	T	T	T	F	F	T	F	F	F	T	F
F	T	T	F	T	T	T	F	F	T	T	T	F	T
F	T	F	F	F	F	T	T	T	F	T	F	T	F
F	F	T											
F	F	F											

Exercise 5 *Complete the last two rows.*

So, we see that a propositional formula can represent a true or a false proposition, depending of the choice of the propositions substituted for the occurring propositional variables, or rather on their truth-values.

1.1.7 Tautologies

Definition 6 *A* ***tautology*** *is a propositional formula that obtains a truth value* `true` *for any assignment of truth values to the occurring variables. Tautologies are also called* ***logically valid formulae.***

Thus, every tautology always renders a true proposition, so it represents a *logical law,* in the same way that the identity $x + y = y + x$ represents a law of arithmetic and, therefore, holds no matter what values we assign to x and y.

Here are a few simple tautologies which represent some important laws of propositional logic:

- $(p \vee \neg p)$: the *law of excluded middle* — every proposition p is either true or false (and in the latter case $\neg p$ must be true);
- $\neg(p \wedge \neg p)$: the *law of non-contradiction* — it cannot be the case that both p and $\neg p$ are true;
- $(p \wedge q) \rightarrow p$: this is always true by the very meaning (and truth table) of $\wedge$;
- likewise, $p \rightarrow (p \vee q)$ is always true by the very meaning and truth table of $\vee$;
- $((p \wedge (p \rightarrow q)) \rightarrow q)$: if p is true and it is true that p implies q, then q is true. This reflects the meaning of the implication.

Checking tautologies with truth tables

How can one determine if a propositional formula A is a tautology? Quite easily: complete the truth-table of A and check if it always takes a truth-value `true`. Let us check some of the examples mentioned above:

p	$\neg p$	$(p \vee \neg p)$	$(p \wedge \neg p)$	$\neg(p \wedge \neg p)$
T	F	T	F	T
F	T	T	F	T

p	q	$(p \rightarrow q)$	$(p \wedge (p \rightarrow q))$	$((p \wedge (p \rightarrow q)) \rightarrow q)$
T	T	T	T	T
T	F	F	F	T
F	T	T	F	T
F	F	T	F	T

The opposite concept of a tautology is a **contradictory formula** — a formula which always takes a truth-value `false`. For instance $(p \wedge \neg p)$ is a contradictory formula.

Question: How are the concepts of tautology and contradictory formula related? How about contradictory and satisfiable formulae?

Checking tautologies by searching for falsifying truth-assignments

Checking tautologies with truth-tables is straightforward, but rather labourious and not too exciting. There is a somewhat streamlined, and more intelligent method, whereby we attempt to show that the formula *is not* a tautology by searching an appropriate **falsifying assignment**, i.e. a combination of truth values of the propositional variables, which renders it false. If we succeed in finding such an assignment, then the formula is indeed not a tautology; if, however, if, however, when exploring a possible case our attempt we reach a state where some variable is required to be both true and false, that is clearly a *contradiction*, meaning that the case we are exploring is impossible, and we abandon that search direction. If *all* possible attempts to produce a falsifying assignment end up with a contradiction, then we have actually *proved that the formula cannot be falsified*, hence it must be a tautology [1].

The systematic search for a falsifying assigment is based in a step-by-step decomposition of the formula by using the truth tables of the propositional connectives occurring in it. Let us demonstrate how this works on a few examples.

To save some writing, we will use **signed formulae** $A : \mathtt{T}$ meaning '*A must be true*' and $A : \mathtt{F}$ meaning '*A must be false*'.

1. Consider $\neg(p \to \neg q) \to (p \vee \neg r)$.

   To falsify it, $\neg(p \to \neg q)$:$\mathtt{T}$ and $p \vee \neg r : \mathtt{F}$ must be the case.

   For the former, $p \to \neg q : \mathtt{F}$, hence $p : \mathtt{T}$ and $\neg q : \mathtt{F}$.

   For the latter, $p : \mathtt{F}$ and $\neg r : \mathtt{F}$.

   Thus, p must be both true and false, which is impossible. Thus, our attempt to falsify the formula has failed, hence it is a tautology.

2. Consider now $\neg p \to \neg(p \vee \neg q)$.

   To make it false, $\neg p : \mathtt{T}$ while $\neg(p \vee \neg q) : \mathtt{F}$ must be the case.

   Therefore $p : \mathtt{F}$ and $p \vee \neg q : \mathtt{T}$.

   For the latter, $p : \mathtt{T}$ or $\neg q : \mathtt{T}$. Let us consider both cases:

   Case 1: $p : \mathtt{T}$. This contradicts $p : \mathtt{F}$.

   Case 2: $\neg q : \mathtt{T}$. Then $q : \mathtt{F}$. In this case we have not reached any contradiction and there is nothing more left to do in order to obtain one. Indeed, we can check that $p : \mathtt{F}$ and $q : \mathtt{F}$ renders the formula false.

3. A contradiction in the truth values can be reached on any *subformula*, not necessarily a variable. For instance, take $(p \vee \neg q) \to (\neg p \to (p \vee \neg q))$. For it to be false, $(p \vee \neg q) : \mathtt{T}$ and $(\neg p \to (p \vee \neg q)) : \mathtt{F}$ must be the case. From the former, $\neg p : \mathtt{T}$ and $(p \vee \neg q) : \mathtt{F}$, which contradicts the latter.

4. Finally, take $((p \wedge \neg q) \to \neg r) \leftrightarrow ((p \wedge r) \to q)$. For it to be false, there are two possible cases:

Case 1 $((p \wedge \neg q) \to \neg r) : \mathtt{T}$ and $((p \wedge r) \to q) : \mathtt{F}$. The latter implies $(p \wedge r) : \mathtt{T}$ and $q : \mathtt{F}$, hence $p : \mathtt{T}$, $q : \mathtt{F}$, $r : \mathtt{T}$, For the former, two sub-cases are possible:

Case 1.1 $\neg r : \mathtt{T}$. Then $r : \mathtt{F}$ which is a contradiction with $r : \mathtt{T}$.

Case 1.2 $(p \wedge \neg q) : \mathtt{F}$. Then:

Case 1.2.1 $p : \mathtt{F}$ — a contradiction with $p : \mathtt{T}$.

Case 1.2.1 $\neg q : \mathtt{F}$. Then $q : \mathtt{T}$ — again a contradiction, now with $q : \mathtt{F}$.

[1] This method of proof is called an *indirect proof* or *proof by contradiction* and will be discussed in more detail later.

Note that the consideration of these sub-cases could have been avoided, if we had noticed that the assignment $p : \texttt{T}$, $q : \texttt{F}$, $r : \texttt{T}$ renders $((p \wedge \neg q) \to \neg r) : \texttt{F}$.

Case 2 $((p \wedge r) \to q) : \texttt{T}$ and $(p \wedge \neg q) \to \neg r) : \texttt{F}$. The former implies $(p \wedge \neg q) : \texttt{T}$ and $\neg r : \texttt{F}$, i.e. $p : \texttt{T}$, $q : \texttt{F}$, $r : \texttt{T}$. Again, we can either notice that this assignment renders $((p \wedge r) \to q) : \texttt{F}$, or consider the cases for $((p \wedge r) \to q) : \texttt{T}$ and see that all lead to a contradiction.

Thus, in neither case can the formula be falsified, so it is a tautology.

This method can be mechanized completely into a kind of *deductive system*, called Semantic Tableau, which will be discussed further in these notes.

1.1.8 Exercises

1. Which of the following are propositions?
   a) $2^3 + 3^2 = 19$
   b) $2^3 + 3^2 = 91$
   c) $2^3 + 3^2 = x$
   d) Will you marry me?
   e) John married on January 1st, 1999.
   f) Mary is not happy.
   g) I told her about John.
   h) If John loves someone else then Mary is not happy.
   i) This sentence is true.

2. If A and B are true propositions and C and D are false ones, determine the truth-values of the following compound propositions:
   a) $A \wedge (B \vee C)$
   b) $(C \to A) \to D$
   c) $C \to (A \to D)$
   d) $\neg(\neg A \vee C) \wedge B$
   e) $\neg(\neg(\neg D \wedge (B \to \neg A)))$
   f) $\neg(C \to A) \vee (C \to D)$
   g) $(C \leftrightarrow \neg B) \vee (A \to \neg A)$

3. Determine the truth-value of the proposition A in each of the following cases:
   a) B and $B \to A$ are true.
   b) $A \to B$ is true and B is false.
   c) $\neg B$ and $A \vee B$ are true.

d) $B \to \neg A$, $\neg B \to \neg C$, and C are true.

e) $\neg C \wedge B$, $C \to (A \vee B)$, and $\neg(A \vee C) \to C$ are true.

(Hint: consider cases for the truth of B and C.)

4. Let P, Q, R, and S be propositions. Using only the truth tables of the propositional connectives and informal reasoning, show that:

a) if the propositions P and $P \to Q$ are true, then Q is true;

b) if $P \to Q$, $Q \to R$ and $P \wedge S$ are true, then $R \wedge S$ is true;

c) if $P \to Q$, $R \vee (S \wedge \neg Q), \neg R$ are true, then P is false;

d) if $\neg R$, $\neg(P \wedge \neg R)$, $\neg P \to \neg Q$ are all true, then Q is false;

e) if the propositions $(P \vee Q) \to R$ and $P \vee R$ are true, then R is true;

(Hint: suppose the contrary.)

5. Let P, Q, R, and S be propositions. Using truth tables, check whether:

a) if $Q \to (R \wedge S)$ is true and $Q \wedge S$ is false, then $R \wedge Q$ is false.

b) if $P \to Q$ and $Q \to (R \vee S)$ are true and $P \to R$ is false, then $\neg R \to S$ is true;

c) if $\neg P \to (\neg Q \vee \neg R)$, $Q \wedge (P \vee R)$ are true, then P is true;

d) if $P \to Q$ and $Q \to (R \vee S)$ are true and $P \to R$ is false, then S is true;

e) if $Q \to (R \wedge S)$ is true and $Q \wedge S$ is false, then Q is false.

(Hint: regard P, Q, R, and S as propositional variables and inspect the truth tables of the respective formulae.)

6. Determine the antecedent and consequent in each of the following implications:

a) If John talks everyone else listens.

b) Everyone else listens if John talks.

c) John talks only if everyone else listens.

d) An integer is positive if its cube is positive.

(Hint: To make it easier, introduce a name for the object in question, e.g. "An integer n is positive if the cube of n is positive.")

e) An integer is positive only if its cube is positive.

f) A function is continuous whenever it is differentiable.

g) The continuity of a function is necessary for it to be differentiable.

h) The continuity of a function is sufficient for it to be differentiable.

7. Which of the following conditions are sufficient, and which are necessary for the truth of "The positive integer n is composite"?

a) "n is divisible by 3."

b) "n is divisible by 6."

c) "n is even."

d) "n has at least two different factors."

e) "n has more than two different factors."

f) "$n = 15$."

g) "n has a factor different from n."

h) "n has a prime factor."

i) "n has a prime factor different from n. "

8. Construct truth-tables for the following propositional formulae and determine which of them (if any) are tautologies, and which are contradictory formulae.

   a) $\neg(p \to \neg p)$

   b) $p \vee (p \to \neg p)$

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

   d) $(p \wedge \neg p) \to q$

   e) $((p \to q) \to p) \to p)$

   f) $\neg p \wedge \neg(p \to q)$

   g) $(p \vee \neg q) \to \neg(q \wedge \neg p)$

   h) $(p \to q) \wedge (q \to r) \wedge \neg(\neg p \vee r)$

   i) $\neg(\neg p \leftrightarrow q) \wedge (r \vee \neg q)$

   j) $\neg((p \wedge \neg q) \to r) \leftrightarrow (\neg(q \vee r) \to \neg p)$

9. On the remote planet Nologic there are two types of intelligent creatures:

   - **truthers**, who always tell the truth, and
   - **liars**, who – you guessed it – always lie.

   It is not possible to distinguish them by appearance, but only by the truth or falsity of the statements they make.

   a) Once a space stranger visited Nologic and met there two inhabitants, P and Q. He asked them: "Is any of you a liar?"."At least one of us is a liar", replied P.

      Can you find out what P and Q are?

   b) Walking about Nologic, the stranger met two other inhabitants, A and B, and asked A, "Is any of you a truther?". "If B is a liar, then I am a liar, too.", replied A.

      What are A and B?

   c) The stranger went on. In the evening, he began to look for a shelter for the night, but was very cautious because he knew that some of the inhabitants were man-eaters and it was not possible to recognize them by appearance. Then he met three inhabitants, C, D and E. He asked C, "How many of you are truthers?". "Flam flim", answered C in her language. "What did she say?", asked the stranger D. "Just one", replied D. "Do not trust D, he is a liar. Come with me, I'm not a man-eater", said E. "No, come with me, I'm not a man-eater", countered D.

What should the stranger do?

10. For many more, entertaining logical puzzles see the marvelous books by Raymond Smullyan: "*What is the name of this book?*", "*The Lady, or the Tiger?*", etc.

1.2 Propositional logical consequence. Valid and invalid propositional inferences.

Logic is *not* concerned with what is true and what is false, but rather with *correctness of argumentation and reasoning.* Here, we will discuss what it means for an argument to be logically correct or not, by formalizing the most fundamental logical concept of *logical consequence* in the simple case of propositional logic.

First, a couple of examples, providing some food for thought.

The famous Greek philosopher Aristotle (384-322 B.C.), who is regarded as the founding father of formal logic, was the first who systematically studied the forms and rules of reasoning. Aristotle studied specific patterns of arguments called *syllogisms.* A typical example of a syllogism is:

$$\frac{\begin{array}{l}\textsf{All logicians are clever.}\\ \textsf{All clever people are rich.}\end{array}}{\textsf{All logicians are rich.}}$$

The way we actually read this is:

If all logicians are clever *and* all clever people are rich, *then* all logicians are rich.

This sounds like a correct piece of reasoning, and it *is* one, *but* (and this is very important to understand) *it does not mean that the conclusion is necessarily true.* (In fact, it is *not*). What then makes it correct?

Here is another one:

$$\frac{\begin{array}{l}\textsf{All natural numbers are integers.}\\ \textsf{Some integers are odd numbers.}\end{array}}{\textsf{Some natural numbers are odd numbers.}}$$

Is this a correct argument? If you think so, judging from the truth of the conclusion (as well as the premises), then consider this: take the same argument and replace the words "natural numbers" by " mice", "integers" by "animals" and "odd numbers" by "elephants". This will not change the logical shape of the argument and, therefore, should not change its logical correctness. The result speaks for itself:

$$\frac{\begin{array}{l}\textsf{All mice are animals.}\\ \textsf{Some animals are elephants.}\end{array}}{\textsf{Some mice are elephants.}}$$

So, *what makes an argument logically correct?* We will answer this question in full later in the course. Now we will provide and discuss the answer for a simple type of logical arguments that only involve propositional reasoning, called *propositional arguments.*

1.2.1 Propositional logical consequence

Definition 7 *A propositional formula B is a* ***logical consequence*** *of the propositional formulae $A_1, \ldots, A_n$, denoted $A_1, \ldots, A_n \models B$, if B is true whenever all $A_1, \ldots, A_n$ are true, that is,*

if every assignment of truth-values to the variables occurring in $A_1, \ldots, A_n, B$ for which the formulae $A_1, \ldots, A_n$ are true, renders the formula B true, too.

If $A_1, \ldots, A_n \models B$, we also say that B ***follows logically from*** *$A_1, \ldots, A_n$, or that $A_1, \ldots, A_n$* ***imply logically*** *B.*

Thus, if $A_1, \ldots, A_n \models B$ then every substitution of propositions for the variables occurring in $A_1, \ldots, A_n, B$ which turns the formulae $A_1, \ldots, A_n$ into true propositions, turns the formula B into a true proposition, too.

In order to check if $A_1, \ldots, A_n \models B$ we can simply complete the truth-tables of $A_1, \ldots, A_n, B$ and check row by row if the following holds: *whenever all formulae $A_1, \ldots, A_n$ have a truth-value* T *in that row, B must have a truth-value* T*, as well*[2]. If that holds in *every row* in the table, then B *does* follow logically from $A_1, \ldots, A_n$; if that fails *in at least one row*, then B *does not* follow logically from $A_1, \ldots, A_n$.

Thus, B *does not* follow logically from $A_1, \ldots, A_n$ if there is *at least one* truth-value assignment that renders all formulae $A_1, \ldots, A_n$ true and B false.

Example 8 1. *(A trivial one.) Any formula B follows logically from any set of formulae that contains B. (Why?)*

2. *For any formulae P and Q, Q follows logically from P and $P \to Q$:*

p	q	p	$p \to q$	q
T	T	T	T	T
T	F	T	F	F
F	T	F	T	T
F	F	F	T	F

Indeed, in every row where the 3rd and 4th entries are T*, the 5th entry is* T*, too.*

Logical consequence can be expressed quite simply in terms of tautologies:

The formula B follows logically from $A_1, \ldots, A_n$ if and only if the formula $(A_1 \wedge \ldots \wedge A_n) \to B$ is a tautology.

Exercise 9 *Prove this.*

Accordingly, the method of checking logical consequences can be streamlined by using semantic tableaux, just like checking tautologies: in order to check if B follows logically from $A_1, \ldots, A_n$ we look for an assignment of truth-values to the variables occurring in $A_1, \ldots, A_n, B$ that renders all $A_1, \ldots, A_n$ true and B false. If we succeed, B *does not* follow logically from $A_1, \ldots, A_n$; otherwise we try to prove that no such assignment is possible by showing that the assumption that it exists leads to a contradiction.

[2]Of course, it is possible for B to be true without all, or any, of $A_1, \ldots, A_n$ being true.

Example 10 *Show that $p \to r$ and $q \to r$ logically imply $(p \vee q) \to r$.*

Method 1 (truth-tables):

p	q	r	$p \to r$	$q \to r$	$p \vee q$	$(p \vee q) \to r$
T	T	T	T	T	T	T
T	T	F	F	F	T	F
T	F	T	T	T	T	T
T	F	F	F	T	T	F
F	T	T	T	T	T	T
F	T	F	T	F	T	F
F	F	T	T	T	F	T
F	F	F	T	T	F	T

In every row where the truth-values of $p \to r$ and $q \to r$ are T, the truth-value of $(p \vee q) \to r$ is T, too.

Method 2 (search for falsifying assignment): Suppose, that for some assignment $(p \to r)$: T, $(q \to r)$: T and $((p \vee q) \to r)$: F. Then $(p \vee q)$: T and r : F , hence p : T or q : T.

Case 1: p : T. Then $(p \to r)$: F - a contradiction.

Case 2: q : T. Then $(q \to r)$: F - again contradiction.

Thus, there is no assignment that falsifies the logical consequence.

1.2.2 Logically sound rules of propositional inference. Logically correct propositional arguments.

Now, we will apply the notion of logical consequence to define and check whether a given propositional argument is logically correct. Let us first introduce the relevant terminology.

Definition 11 *A* ***rule of propositional inference (inference rule****, for short) is a scheme:*

$$\frac{P_1, \dots, P_n}{C}$$

where $P_1, \dots, P_n, C$ are propositional formulae. The formulae $P_1, \dots, P_n$ are called ***premises*** *of the inference rule, and C is its* ***conclusion****.*

An ***instance*** *of an inference rule is obtained by uniform substitution of concrete propositions for the variables occurring in all formulae of the rule. Every such instance is called a* ***propositional inference****, or a* ***propositional argument*** *based on that rule.*

Definition 12 *An inference rule is* ***logically sound*** *if its conclusion follows logically from the premises. A propositional argument is* ***logically correct*** *if it is an instance of a logically sound inference rule.*

Example 13

1. *The following inference rule*
$$\frac{p,\ p \to q}{q}$$
is sound, as we have already seen. This rule is known as the Detachment rule*, or* ***Modus Ponens****, and is very important in some logical deductive systems,*

   Therefore, the inference

   Julia is singing.
   If Julia is singing, then Julia is happy.
   ___
   Julia is happy.

   is logically correct, being an instance of that inference rule.

2. *The inference rule*
$$\frac{p,\ q \to p}{q}$$
is not sound*: if p is true and q is false, then both premises are true, while the conclusion is false.*

   Therefore the inference

   2 plus 2 equals 4.
   If 5 is greater than 3, then 2 plus 2 equals 4.
   ___
   5 is greater than 3.

   is not logically correct (despite the truth of the conclusion) being an instance of an unsound rule.

Some remarks are in order here.

- It is very important to realize that the logical correctness of an inference *does not always guarantee the truth of the conclusion, but only when all premises are true.* In other words, if at least one premise of a logically correct inference is false, then the conclusion may be false, too. For instance, the inference

  5 divides 6.
  If 5 divides 6, then 5 divides 11.
  ___
  5 divides 11.

  is logically correct (being an instance of the rule Modus Ponens) in spite of the falsity of the conclusion, and this does not contradict our idea of a logical correctness of an inference, because the first premise is false. (What about the second premise?)

- Conversely, if the conclusion of an inference happens to be true, this does not necessarily mean that the inference is logically correct, as in the second example given above.

- Moreover, it may happen that the truth of the premises of an inference *does imply* the truth of the conclusion, and yet the inference is not *logically* correct. For instance, the correctness of the inferences

$$\frac{\textsf{Today is Monday.}}{\textsf{Tomorrow will be Tuesday.}}$$

or

$$\frac{a = 2,\ a + b = 5}{b = 3}$$

is based *not* on logical consequence but, in the first case, on the common fact that Tuesday always follows Monday, and in the second case on some laws of arithmetic. Indeed, the first inference is based on the rule

$$\frac{p}{q}$$

and the second one (though the statements in that rule are not real propositions) on

$$\frac{p,\ q}{r}$$

both of which are clearly unsound.

To summarize, the meaning and importance of logically correct inferences is that only such inferences guarantee that *if* all premises are true, *then* the conclusions will be true, too. That is why, only logically correct inferences are safe to be employed in our reasoning.

Let us now look at a couple of more examples.

1. The inference rule

   $$\frac{q,\ p \vee \neg q}{p}$$

   is logically sound. You can check this in two ways: by applying the definition or by showing that the corresponding formula

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

   is a tautology.

   Consequently, the inference

   $$\frac{\begin{array}{l}\textsf{Josephine is singing or Josephine does not feel happy.}\\ \textsf{Josephine feels happy.}\end{array}}{\textsf{Josephine is singing.}}$$

   is logically correct, being an instance of the rule above.

2. Now, take the argument

   $$\frac{\begin{array}{l}\textsf{If } a \textsf{ divides } b \textsf{ or } a \textsf{ divides } c \textsf{ , then } a \textsf{ divides } bc.\\ a \textsf{ divides } bc.\\ a \textsf{ does not divide } b.\end{array}}{\textsf{Therefore } a \textsf{ divides } c.}$$

   where a, b, c are certain integers. This argument is an instance of the following inference rule:

   $$\frac{(p \vee q) \to r,\ r,\ \neg p}{q}.$$

   Let us see if we can invalidate this rule. For that we need an assignment such that $((p \vee q) \to r) : \mathtt{T}, r : \mathtt{T}, \neg p : \mathtt{T}$, hence $p : \mathtt{F}$, and $q : \mathtt{F}$. Indeed, the assignment $p : \mathtt{F}, q : \mathtt{F}, r : \mathtt{T}$ renders all premises true and the conclusion false. (Check this.)

So, the rule is not logically sound, hence the argument above is not logically correct.

1.2.3 Fallacies of the implication

As an application we will analyze some very common forms of correct and incorrect reasoning related to implications. Given the implication

$$A \to B$$

we can form the so-called *derivative implications*:

- the *converse* of $A \to B$: $B \to A$;
- the *inverse* of $A \to B$: $\neg A \to \neg B$;
- the *contrapositive* of $A \to B$: $\neg B \to \neg A$.

Now, suppose we know that $A \to B$ is true. What can we say about the truth of its derivatives? To answer that question, look at the inferences

$$\frac{A \to B}{B \to A}, \quad \frac{A \to B}{\neg A \to \neg B}, \quad \frac{A \to B}{\neg B \to \neg A}$$

Exercise 14 *Show that the first two of these inferences are incorrect, while the third one is correct.*

Therefore, the truth of an implication $A \to B$ *only implies the truth of its contrapositive*, but not the truth of the converse, or inverse. These are mistakes which people often make, respectively called *the fallacy of the converse implication* and *the fallacy of the inverse implication*. For example, the truth of the implication "If it has just rained, then the tennis court is wet." does not imply that either of "If the tennis court is wet, then it has just rained" or "If it has not just rained, then the tennis court is not wet" is true (for someone may have just watered the court on a clear sunny day), but it certainly implies that " If the court is not wet, then it has not just rained".

In fact, you can easily check that the argument

$$\frac{\neg B \to \neg A}{A \to B}$$

is logically correct, too. This rule is the basis of the method of *proof by contraposition*, which will be discussed later.

1.2.4 Exercises

1. Prove that $A_1, \ldots, A_n \models B$ if and only if $\models (A_1 \wedge \ldots \wedge A_n) \to B$.
2. Using truth-tables, check if the following inference rules are valid.

   a) $$\frac{p \to q,\ \neg q \vee r,\ \neg r}{\neg p}$$

   b) $$\frac{p \to q,\ p \vee \neg r,\ \neg r}{\neg q \vee r}$$

   c) $$\frac{\neg p \to \neg q,\ q,\ \neg(p \wedge \neg r)}{r}$$

   d) $$\frac{((p \wedge q) \to r),\ \neg(p \to r)}{q \to r}$$

3. Write down the inference rules on which the following arguments are based and check their logical validity, using truth tables.

   a) In the following argument, x is a certain number.

   x is greater than 3.

   ---
   x is greater than or equal to 3.

   b) In the following argument, a is a certain number.

   If a is greater than -1, then a is greater than -2.
   a is not greater than -2.

   ---
   a is not greater than -1.

   c)
   If the triangle ABC has a right angle, then it is not equilateral.
   The triangle ABC does not have a right angle.

   Therefore, the triangle ABC is equilateral.

   d)
   If Thomas likes logic, then he is clever.
   If Thomas is clever, then he is rich.

   Therefore, if Thomas likes logic, then he is rich.

   e) In the following argument n is a certain integer.

   If n is divisible by 2 and n is divisible by 3, then n is divisible by 6.
   If n is divisible by 6, then n is divisible by 2.
   n is not divisible by 3.

   Therefore, n is not divisible by 6.

4. For each of the following implications construct the converse, inverse and the contrapositive.

   a) If a is greater than -1, then a is greater than -2.

   b) If x is divisible by 6 then x is not prime.

   c) x is positive only if its square is positive.

   d) The triangle ABC is equilateral whenever its medicentre and orthocentre coincide.

   e) For the function f to be continuous it is sufficient that it is differentiable.

   f) For the function f not to be differentiable it is sufficient that it is discontinuous.

   g) For the integer n to be prime it is necessary that it is not divisible by 10.

1.3 Propositional logical equivalence. Negation normal form.

1.3.1 Logically equivalent propositional formulae

Definition 15 *Propositional formulae A and B are**logically equivalent**, denoted by $A \equiv B$, if for every assignment of truth-values to the variables occurring in them they obtain the same truth values.*

Being a little imprecise (you'll see why), we can say that A and B are *logically equivalent* if they have the same truth-tables.

- Every tautology is equivalent to $\top$. For example, $p \vee \neg p \equiv \top$.
- Every contradiction is equivalent to $\bot$. For example, $p \wedge \neg p \equiv \bot$.
- $\neg\neg p \equiv p$: a double negation of a proposition is equivalent to the proposition itself.
- $\neg(p \wedge q) \equiv (\neg p \vee \neg q)$ and $\neg(p \vee q) \equiv (\neg p \wedge \neg q)$. These are known as *De Morgan's laws* [3]. Let us check the first one using the simplified version of truth-tables:

p	q	$\neg$	$(p$	$\wedge$	$q)$	$(\neg$	p	$\vee$	$\neg$	$q)$
T	T	F	T	T	T	F	T	F	F	T
T	F	T	T	F	F	F	T	T	T	F
F	T	T	F	F	T	T	F	T	F	T
F	F	T	F	F	F	T	F	T	T	F

- $(p \wedge (p \vee q)) \equiv (p \wedge p)$. There is a small problem here: formally, the truth-tables of these formulae are *not* the same, as the first one contains two variables (p and q) while the second one contains only p. However, we can always think that q occurs *vacuously* in the second formula and include it in its truth-table:

p	q	$(p$	$\wedge$	$(p$	$\vee$	$q))$	$(p$	$\wedge$	$p)$
T	T	T	T	T	T	T	T	T	T
T	F	T	T	T	T	F	T	T	T
F	T	F	F	F	T	T	F	F	F
F	F	F	F	F	F	F	F	F	F

Checking logical equivalence can be streamlined, just like checking logical validity and consequence, by systematic search for a falsifying assignment, viz.: in order to check if $A \equiv B$ we try to construct a truth assignment to the variables occurring in A and B which renders one of them true while the other — false. If such assignment exists, the formulae are *not* logically equivalent, otherwise they are. For example, let us check the second De Morgan's law: $\neg(p \vee q) \equiv (\neg p \wedge \neg q)$. There are two possibilities for an assignment to falsify that equivalence:

(i) $\neg(p \vee q) : \mathtt{T}, (\neg p \wedge \neg q) : \mathtt{F}$. Then $p \vee q : \mathtt{F}$ hence $p : \mathtt{F}$, and $q : \mathtt{F}$, but then $(\neg p \wedge \neg q) : \mathtt{T}$ — a contradiction.

[3] After the British mathematician and logician Augustus De Morgan (1806-1871) who, together with the mathematician George Boole (1815-1864), pioneered in applying algebraic methods to the study of logic, one and a half centuries ago. De Morgan was the first to note these logical equivalences.

(ii) $\neg(p \vee q) : \mathtt{F}, (\neg p \wedge \neg q) : \mathtt{T}$. Then $\neg p : \mathtt{T}$ and $\neg q : \mathtt{T}$ hence $p : \mathtt{F}$, and $q : \mathtt{F}$, but then $\neg(p \vee q) : \mathtt{T}$ — a contradiction again.

Thus, there is no falsifying assignment, hence the equivalence holds.

Some basic properties of logical equivalence

Caution: Do not confuse the logical connective $\leftrightarrow$ and logical equivalence between formulae. These are different things: the former is a logical connective, a symbol in our *object language*, whereas the latter is a relation between formulae, statement in our *metalanguage.* However, as we show below, there is a simple relation between them.

1. Logical equivalence is reducible to logical validity:
$$A \equiv B \text{ iff } \models A \leftrightarrow B.$$
Indeed, both mean the same: that A and B always take the same truth-values.
2. Logical equivalence is reducible to logical consequence:
$$A \equiv B \text{ iff } A \models B \text{ and } B \models A$$
3. $\equiv$ is an **equivalence relation**, that is, for every formulae A, B, C:
   a) *reflexive*: $A \equiv A$;
   b) *symmetric*: if $A \equiv B$ then $B \equiv A$;
   c) *transitive*: if $A \equiv B$ and $B \equiv C$ then $A \equiv C$.
4. Moreover, $\equiv$ is a *congruence* with respect to the propositional connectives, that is:
   a) if $A \equiv B$ then $\neg A \equiv \neg B$;
   b) if $A_1 \equiv B_1$ and $A_2 \equiv B_2$ then $(A_1 \bullet A_2) \equiv (B_1 \bullet B_2)$, where $\bullet \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$.

Thus, logical equivalence between propositional formulae can be treated just like equality between algebraic expressions.

In particular, the following property of **equivalent replacement** holds. For any propositional formulae A, B, C and a propositional variable p, presumably occurring in C, if $A \equiv B$ then $C(A/p) \equiv C(B/p)$, where $C(X/p)$ is the result of simultaneous substitution of al occurrences of p by X.

1.3.2 Some important logical equivalences

Here we will list several groups of important and useful logical equivalences. Verifying them all are easy, but useful exercises.

Equivalences expressing algebraic laws for the logical connectives

We begin with some important logical equivalences which are used e.g., for equivalent transformations of propositional formulae to so called *conjunctive and disjunctive normal forms* that we will introduce later.

- *Idempotency:* $p \wedge p \equiv p$; $p \vee p \equiv p$.
- *Commutativity:* $p \wedge q \equiv q \wedge p$; $p \vee q \equiv q \vee p$.
- *Associativity:* $(p \wedge (q \wedge r)) \equiv ((p \wedge q) \wedge r)$; $(p \vee (q \vee r)) \equiv ((p \vee q) \vee r)$.

  This property allows us to omit the parentheses in multiple conjunctions and disjunctions.
- *Absorption:* $p \wedge (p \vee q) \equiv p$; $p \vee (p \wedge q) \equiv p$;
- *Distributivity:* $p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r)$; $p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$.

Equivalences mutually expressing logical connectives

The following logical equivalences express logical connectives in terms of others:

- $\neg A \equiv A \to \bot$

  (We sometimes use this equivalence in colloquial expressions, such as "If this is true then I can fly", meaning to say: "This cannot be true".)
- $A \leftrightarrow B \equiv (A \to B) \wedge (B \to A)$.

  This equivalence allows us to consider the biconditional as definable connective, which we will do often.
- $A \vee B \equiv \neg(\neg A \wedge \neg B)$;
- $A \wedge B \equiv \neg(\neg A \vee \neg B)$;
- $A \to B \equiv \neg A \vee B \equiv \neg(A \wedge \neg B)$
- $A \vee B \equiv \neg A \to B$;
- $A \wedge B \equiv \neg(A \to \neg B)$.

Thus, we see that each of $\wedge, \vee, \to$ can be expressed by means of any other of them, using negation.

Some simplifying equivalences

Other useful logical equivalences can be used to simplify formulae:

- $A \vee \neg A \equiv \top$; $A \wedge \neg A \equiv \bot$
- $A \wedge \top \equiv A$; $A \wedge \bot \equiv \bot$
- $A \vee \top \equiv \top$; $A \vee \bot \equiv A$
- $\neg A \to \neg B \equiv B \to A$ (Every implication is equivalent to its contrapositive.)

Negating propositional formulae. Negation normal form.

In mathematical and other arguments we sometimes have to negate a formalized statement and then use the result in further reasoning. For that, it is useful to transform, up to logical equivalence, the formula formalizing the statement in **negation normal form**, where negation may only occur in front of propositional variables. Such transformation can be done by step-by-step importing all occurrences of negations inside the other logical connectives, using the following equivalences, some of which we already know:

- $\neg\neg A \equiv A$,
- $\neg(A \land B) \equiv \neg A \lor \neg B$,
- $\neg(A \lor B) \equiv \neg A \land \neg B$,
- $\neg(A \to B) \equiv A \land \neg B$,
- $\neg(A \leftrightarrow B) \equiv (A \land \neg B) \lor (B \land \neg A)$.

Example 16 *Equivalent transformation to negation normal form:*
$\neg((A \lor \neg B) \to (\neg C \land D))$
$\equiv (A \lor \neg B) \land \neg(\neg C \land D)$
$\equiv (A \lor \neg B) \land (\neg\neg C \lor \neg D)$
$\equiv (A \lor \neg B) \land (C \lor \neg D)$.

1.3.3 Exercises

1. Verify the following logical laws:
   a) *Idempotency:* $p \land p \equiv p, \ \ p \lor p \equiv p$
   b) *Commutativity:* $p \land q \equiv q \land p, \ \ p \lor q \equiv q \lor p$
   c) *Associativity:* $(p \land (q \land r)) \equiv ((p \land q) \land r), \ \ (p \lor (q \lor r)) \equiv ((p \lor q) \lor r)$
   d) *Absorption:* $p \land (p \lor q) \equiv p, \ \ p \lor (p \land q) \equiv p$
   e) *Distributivity:* $p \land (q \lor r) \equiv (p \land q) \lor (p \land r), \ \ p \lor (q \land r) \equiv (p \lor q) \land (p \lor r)$
   f) *De Morgan's laws* $\neg(p \land q) \equiv (\neg p \lor \neg q), \ \ \neg(p \lor q) \equiv (\neg p \land \neg q)$
2. Prove the following logical equivalences:
   a) $\neg p \lor q \equiv p \to q$
   b) $p \leftrightarrow q \equiv (p \to q) \land (q \to p)$
   c) $\neg(p \to q) \equiv p \land \neg q$,
   d) $\neg(p \leftrightarrow q) \equiv (p \land \neg q) \lor (q \land \neg p)$.
   e) $\neg(p \leftrightarrow q) \equiv \neg p \leftrightarrow q$
   f) $(p \to q) \land (p \to r) \equiv p \to (q \land r)$
   g) $p \leftrightarrow (q \leftrightarrow r) \equiv (p \leftrightarrow q) \leftrightarrow r$
3. Determine which of the following pairs of formulae are logically equivalent:
   a) $p \to q, \ q \to p$
   b) $p \to q, \ \neg p \to \neg q$

c) $p \to \neg q,\ q \to \neg p$

d) $\neg(p \to \neg q),\ p \wedge q$

e) $((p \to q) \to q) \to q,\ p \vee q$

f) $(p \to r) \wedge (q \to r),\ (p \wedge q) \to r$

g) $(p \to r) \vee (q \to r),\ (p \vee q) \to r$

h) $(p \to q) \vee (p \to r),\ p \to (q \vee r)$

i) $((p \wedge q) \to r),\ (p \to r) \vee (q \to r)$

j) $((p \vee q) \to r),\ (p \to r) \wedge (q \to r)$

k) $p \to (q \to r),\ (p \to q) \to r$

l) $p \leftrightarrow (q \leftrightarrow r),\ q \leftrightarrow (p \leftrightarrow r)$

m) $p \to (q \to r),\ (p \to q) \to (p \to r)$

4. Negate each of the following propositional formulae and transform the result to an equivalent formula in a negation normal form.

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

b) $(p \to \neg q) \to p$

c) $p \to (\neg q \to p)$

d) $(p \leftrightarrow \neg q) \to \neg r$

e) $p \to (\neg q \leftrightarrow r)$

1.4 The concept and use of deductive systems

The most important concept in Logic is logical consequence, which is the foundation of logically correct inferences. While establishing logical validity and logical consequence in propositional logic is conceptually and technically easy (though, possibly computationally expensive), this is not so anymore for the first-order logic, nor for most of the non-classical logics. In fact, it is usually an *infinite task*, as it generally requires checking infinitely many possible 'models', rather than a finite number of simple truth assignments. Thus, a different approach for establishing logical validity and consequence, not based on the semantic definition, becomes practically necessary. Such different approach is provided by the notion of *deductive system* - a formal, mechanical (or, at least mechanizable) procedure for *derivation (inference, deduction)* of formulae, or 'sequents' of formulae, by applying precise *inference rules* and possibly using some formulae that are postulated as derived, called *axioms*. The underlying idea of deductive systems is to *simulate* and *substitute* the semantic notion of logical consequence with the formal, syntactic notion of *deductive consequence*, which is based only on the syntactic shape of the formulae to which inference rules are applied, but not on their meaning. Thus, at least theoretically, a person with no knowledge and understanding of the meaning of logical formulae and logical consequence, or even a suitably programmed computer, should be able to do successful derivations in a given deductive system.

The idea of deductive systems goes back to the antiquity. Perhaps the first prototype of a deductive system can be found in Euclid's *Elements* where he provides a systematic development of elementary geometry, based on several simple assumptions about points and lines (such as "Every two different points determine exactly one line", "For every line there is a point not belonging to that line", etc.). Using these and some (informal) logical reasoning, other geometric facts are derived, and thus the entire body of Euclidean geometry is eventually built. The *logical* concept of deductive system gradually emerged much later, notably through the ideas of Leibniz for a *characteristica universalis* (universal language) and a *calculus ratiocinator* (calculus of reasoning). The first really formal deductive systems, however, were only constructed in the late 19th century by Gottlob Frege, for the prototype of modern first-order logic in his book 'Begriffsschrift', David Hilbert, who re-worked Euclid's system of geometry developed in the "Elements" into a rigorous and mathematically precise treatment that resulted in his book "Foundations of Geometry", Giuseppe Peano, for a formal system of arithmetic, still known as Peano axiomatic system. The concept of formal deductive system was further developed by Bertrand Russell and Alfred North Whitehead in their book 'Principia Mathematica', and by David Hilbert and Wilhelm Ackermann in their book 'Principles of Mathematical Logic' – both books being most influential for the development of logic and foundations of mathematics in the first half of the 20th century. In particular, David Hilbert, one of the most influential mathematicians of the late 19th and early 20th century, vigorously promoted the idea of building the whole body of mathematics as a formal *axiomatic system* – a deductive system mainly based on axioms and on very few and simple rules of inference. Thus, Hilbert was the leading proponent of the development of the *formal, axiomatic approach* in mathematics, meant to replace the semi-formal notion of mathematical proof by the completely formalized notion of derivation in an axiomatic system, thus freeing mathematics from paradoxes and other possible contradictions and putting it on sound logical foundations.[4]

[4]In the early 1930s it turned out, from the groundbreaking works of Kurt Gödel that Hilbert's idea was only

There are different types of deductive systems, some of which we will present here, but they all share several common aspects and principles:

- A deductive system works within a **formal logical language**, where derivations are performed on formal expressions, called **formulae** (or, 'sequents', being finite sequences of formulae) which are special words – finite strings of symbols – in the formal language, having precise **syntax**[5] In this section we discuss deductive systems acting on propositional formulae, and later – on first-order formulae.
- The main component of a deductive system is the notion of **derivation (inference, deduction) from a given set of assumptions**, based on a set of precisely specified **rules of inference**. The idea, again, is that by applying systematically these rules, one can **derive (infer, deduce)** formulae from other formulae that are already derived, or *assumed* as given, and called **assumptions (premisses)**. In addition, axiomatic systems also allow for an initial set of formulae, called **axioms**, to be accepted as derived without applying any rules of inference. Thus, the axioms can always be used as premises in derivations.

  If a formula A can be derived from a set of assumptions Γ in a deductive system $\mathcal{D}$, we denote this by
  $$\Gamma \vdash_{\mathcal{D}} A$$
  and say that A **is a deductive consequence from Γ in $\mathcal{D}$**.

  In particular, if $\Gamma = \varnothing$ we write
  $$\vdash_{\mathcal{D}} A$$
  and say that A is a **theorem** of $\mathcal{D}$.
- A very important aspect of deductive systems is that derivations in them are *completely mechanical procedures*, that do not require any intelligence or any understanding of the meaning of the formulae or rules involved; in fact such meaning need not be specified at all. Thus, derivations in a given deductive system can be performed by a mechanical device, such as a computer, without any human intervention, as long as the axioms and rules of inference of the deductive system have been programmed into it.
- While deductive systems are not concerned with the meaning *(semantics)* of the formulae they derive, usually they are designed with the purpose to derive *valid, and only valid* logical consequences from the assumptions. A deductive system with this property is called **sound**, or **correct.** In particular, every theorem of such a deductive system must always be true, i.e., in the case of propositional logic it must be a tautology.

  Formally, a deductive system $\mathcal{D}$ is sound (correct) for a given logical semantics if $\mathcal{D}$ can *only* derive logically valid consequences, i.e.:
  $$A_1, \ldots, A_n \vdash_{\mathcal{D}} C \quad \Longrightarrow \quad A_1, \ldots, A_n \models C$$
  In particular:
  $$\vdash_{\mathcal{D}} C \quad \Longrightarrow \models C$$

partly realizable, for purely logical validity and consequence in first-order logic, but it was not realizable for 'mathematical' consequence, even in relatively simple mathematical systems such as arithmetic.

[5]In this aspect a deductive system is much like a programming language.

- A deductive system $\mathcal{D}$ is **complete** for a given logical semantics if $\mathcal{D}$ can derive *every* valid logical consequence, i.e.:

  $$A_1, \dots, A_n \models C \implies A_1, \dots, A_n \vdash_{\mathcal{D}} C$$

  In particular:

  $$\models C \implies \vdash_{\mathcal{D}} C,$$

  that it, a complete deductive system can derive *every* logically valid formula.

  A deductive system $\mathcal{D}$ is **adequate** for a given semantics if it is both sound and complete for it, i.e.:

  $$A_1, \dots, A_n \models C \iff A_1, \dots, A_n \vdash_{\mathcal{D}} C$$

  In particular: $\models C \iff \vdash_{\mathcal{D}} C$.

The most popular types of deductive systems for classical logic are: *axiomatic systems*, *sequent calculi*, *semantic tableau*, *natural deduction*, and *resolution*. Sequent calculi are easily reducible to semantic tableau or natural deduction, so we will not discuss them in this course.

1.5 Semantic tableau

Here we are going to systematize the idea of systematic search for a falsifying truth-assignment, intuitively outlined earlier, into a formal deductive system. Recall, that in order to prove the consequence $A_1, \ldots, A_n \models B$, the method of semantic tableau requires to show that there is no truth assignment which falsifies it, which is equivalent to showing that the formulae $A_1, \ldots, A_n$ and $\neg B$ *cannot be satisfied simultaneously.*

The deductive system of Semantic Tableau **ST** is based on **formula-decomposition rules** which reduce the truth or falsity of a formula to the truth or falsity of its main subformulas. These rules can be extracted from the truth tables of the propositional connectives as follows:

1. $\neg$:
   a) For $\neg A$ to be true, A must be false.
   b) For $\neg A$ to be false, A must be true.
2. $\wedge$:
   a) For $A \wedge B$ to be true, A must be true *and* B must be true.
   b) For $A \wedge B$ to be false, A must be false *or* B must be false.
3. $\vee$:
   a) For $A \vee B$ to be true, A must be true *or* B must be true.
   b) For $A \vee B$ to be false, A must be false *and* B must be false.
4. $\rightarrow$:
   a) For $A \rightarrow B$ to be true, A must be false *or* B must be true.
   b) For $A \rightarrow B$ to be false, A must be true *and* B must be false.
5. $\leftrightarrow$:
   a) For $A \leftrightarrow B$ to be true, $A \rightarrow B$ must be true *and* $B \rightarrow A$ must be true.
   b) For $A \leftrightarrow B$ to be false, $A \rightarrow B$ must be false *or* $B \rightarrow A$ must be false.

These rules can be formalized further as follows:

Rules for Semantic Tableau in propositional logic

Non-branching rules (α-rules)	**Branching rules (β-rules)**
($\land\mathtt{T}$) $\begin{array}{c} A \land B : \mathtt{T} \\ \downarrow \\ A : \mathtt{T}, B : \mathtt{T} \end{array}$	($\land\mathtt{F}$) $\begin{array}{cc} \multicolumn{2}{c}{A \land B : \mathtt{F}} \\ \swarrow & \searrow \\ A : \mathtt{F} & B : \mathtt{F} \end{array}$
($\lor\mathtt{F}$) $\begin{array}{c} A \lor B : \mathtt{F} \\ \downarrow \\ A : \mathtt{F}, B : \mathtt{F} \end{array}$	($\lor\mathtt{T}$) $\begin{array}{cc} \multicolumn{2}{c}{A \lor B : \mathtt{T}} \\ \swarrow & \searrow \\ A : \mathtt{T} & B : \mathtt{T} \end{array}$
($\to\mathtt{F}$) $\begin{array}{c} A \to B : \mathtt{F} \\ \downarrow \\ A : \mathtt{T}, B : \mathtt{F} \end{array}$	($\to\mathtt{T}$) $\begin{array}{cc} \multicolumn{2}{c}{A \to B : \mathtt{T}} \\ \swarrow & \searrow \\ A : \mathtt{F} & B : \mathtt{T} \end{array}$
($\neg\mathtt{T}$) $\begin{array}{c} \neg A : \mathtt{T} \\ \downarrow \\ A : \mathtt{F} \end{array}$	
($\neg\mathtt{F}$) $\begin{array}{c} \neg A : \mathtt{F} \\ \downarrow \\ A : \mathtt{T} \end{array}$	

Using these rules we can search *systematically* for an assignment falsifying a formula. In the process of this systematic application of the rules we build a 'search tree' called **tableau,** as follows:

If we are to derive the consequence $A_1, \ldots, A_n \models B$ we place the signed formulae

$$A_1 : \mathtt{T}, \ldots, A_n : \mathtt{T}, B : \mathtt{F}$$

at the **root** of the tree and then extend it downwards by applying the decomposition rules and adding one (for the α-rules) or two (for the β-rules) successor nodes labelled with signed formulae which that rule introduces. We repeat that procedure along every branch of the tableau until:

- either a **contradictory pair of signed formulae** of the type $A : \mathtt{T}$, $A : \mathtt{F}$ appears on the branch, in which case we declare that branch **closed** (meaning that no assignment can ever be found which satisfies all signed formule on that branch), mark it by $\times$ and do not extend it any further,

  or

- no *new* applications of decomposition rules are possible on that branch, i.e. every signed formula appearing on the branch has already been decomposed further on that branch. Then we say that the branch is **saturated.** If a saturated branch is not closed yet, we declare it **open** and mark it by $\bigcirc$.

Note that an open branch defines an assignment for all variables occurring on that branch as follows: a variable p is true under that assignment if $p : \mathtt{T}$ appears on the branch, and is false otherwise (in which case, $p : \mathtt{F}$ is certain to appear on the branch). It is easy to check that such assignment will satisfy *all* signed formulae on the branch. That can be done step-by-step by going up the branch. Eventually, all signed formulae at the root will be satisfied, which will falsify the logical consequence $A_1, \ldots, A_n \models B$ and will thus prove it invalid.

On the other hand, if all branches of the tableau close, then the entire tableau is declared **closed.** That means that the systematic search for an assignment satisfying the formulae at the root has failed. Then we say that $A_1, \ldots, A_n \models B$ is **derived** in **ST**, which will be denoted by $A_1, \ldots, A_n \vdash_{\mathbf{ST}} B$. In particular, if $\varnothing \vdash_{\mathbf{ST}} B$ then we say that B is a **theorem** of **ST**.

In the long run, it can be proved that Semantic Tableau is *sound and complete*, i.e.:

$A_1, \ldots, A_n \vdash_{\mathbf{ST}} B$ *if and only if* $A_1, \ldots, A_n \models B$ *is valid.*

In particular, $\vdash_{\mathbf{ST}} B$ *if and only if* B *is a tautology.*

Let us demonstrate the method of semantic tableau on some of the examples we did informally in Section 1:

1. Show that $\neg(p \to \neg q) \to (p \vee \neg r)$ is a tautology:

$$\begin{array}{c}
\neg(p \to \neg q) \to (p \vee \neg r) : \mathtt{F} \\
\downarrow \\
\neg(p \to \neg q) : \mathtt{T}, (p \vee \neg r) : \mathtt{F}^1 \\
\downarrow \\
p \to \neg q : \mathtt{F} \\
\downarrow \\
p : \mathtt{T}, \neg q : \mathtt{F} \\
\downarrow \\
q : \mathtt{T} \\
\downarrow^1 \\
p : \mathtt{F}, \neg r : \mathtt{F} \\
\times
\end{array}$$

Note that no branching rules were applicable here, so the tableau consists of only one branch, which closes because the complementary pair $p : \mathtt{T}, p : \mathtt{F}$ appears on it. The superscript ^{1} indicates where the decomposition rule is applied to the formula $(p \vee \neg r) : \mathtt{F}$. This index is not needed, it has only been included to help reading the tableau, and will not be used further.

Thus, $\vdash_{\mathbf{ST}} \neg(p \to \neg q) \to (p \vee \neg r)$ hence $\models \neg(p \to \neg q) \to (p \vee \neg r)$.

2. Check if $\neg p \models \neg(p \lor \neg q)$.

$$\begin{tikzcd}
 & \neg p : \mathtt{T}, \neg(p \lor \neg q) : \mathtt{F} \arrow[d] & \\
 & p : \mathtt{F} \arrow[d] & \\
 & p \lor \neg q : \mathtt{T} \arrow[dl] \arrow[dr] & \\
p : \mathtt{T} & & \neg q : \mathtt{T} \arrow[d] \\
\times & & q : \mathtt{F} \\
 & & \bigcirc
\end{tikzcd}$$

The left-hand branch closes, but the right-hand one does not, and no more decomposition rules are applicable on it, so it is open, and therefore the whole tableau is open, which means that $\neg p \not\models \neg(p \lor \neg q)$. Indeed, the right-hand branch provides a falsifying valuation for that consequence: $p : \mathtt{F}, q : \mathtt{F}$. (check it!)

3. Check if $(\neg p \to q), \neg r \models p \lor \neg q$:

$$\begin{tikzcd}
 & (\neg p \to q) : \mathtt{T}, \neg r : \mathtt{T}, p \lor \neg q : \mathtt{F} \arrow[d] & \\
 & r : \mathtt{F} \arrow[d] & \\
 & p : \mathtt{F}, \neg q : \mathtt{F} \arrow[d] & \\
 & q : \mathtt{T} \arrow[dl] \arrow[dr] & \\
\neg p : \mathtt{F} \arrow[d] & & q : \mathtt{T} \\
p : \mathtt{T} & & \bigcirc \\
\times & &
\end{tikzcd}$$

Again, the tableau is open, and a falsifying valuation is provided by the open branch: $p : \mathtt{F}, q : \mathtt{T}, r : \mathtt{F}$. Note that we chose to first decompose the subformulas $\neg r : \mathtt{T}$ and $(p \lor \neg q) : \mathtt{F}$ because they are non-branching (α-formulas) and then $\neg p \to q : \mathtt{T}$ which is branching (β-formula). This does not make a difference in the outcome of the tableau but saves some work, so it is a generally preferrable strategy.

4. Finally, check if $((p \land \neg q) \to \neg r) \leftrightarrow ((p \land r) \to q)$ is a tautology. We could have introduces tableau rules for $\leftrightarrow$, but it is not really necessary. Instead, we consider $A \leftrightarrow B$ as an abbreviation of $(A \to B) \land (B \to A)$.

$$\begin{tikzcd}[column sep=small]
 & & & ((p \wedge \neg q) \to \neg r) \leftrightarrow ((p \wedge r) \to q) : \mathtt{F} \arrow[d] & & \\
 & & & (((p \wedge \neg q) \to \neg r) \to ((p \wedge r) \to q)) \wedge (((p \wedge r) \to q) \to ((p \wedge \neg q) \to \neg r)) : \mathtt{F} \arrow[dll] \arrow[dr] & & \\
 & ((p \wedge \neg q) \to \neg r) \to ((p \wedge r) \to q) : \mathtt{F} \arrow[d] & & & ((p \wedge r) \to q) \to ((p \wedge \neg q) \to \neg r) : \mathtt{F} \arrow[d] & \\
 & (p \wedge \neg q) \to \neg r : \mathtt{T}, (p \wedge r) \to q : \mathtt{F} \arrow[d] & & & (p \wedge r) \to q : \mathtt{T}, (p \wedge \neg q) \to \neg r : \mathtt{F} \arrow[d] & \\
 & p \wedge r : \mathtt{T}, q : \mathtt{F} \arrow[d] & & & p \wedge \neg q : \mathtt{T}, \neg r : \mathtt{F} \arrow[d] & \\
 & p : \mathtt{T}, r : \mathtt{T} \arrow[dl] \arrow[dr] & & & p : \mathtt{T}, \neg q : \mathtt{T} \arrow[d] & \\
p \wedge \neg q : \mathtt{F} \arrow[d] \arrow[dr] & & \neg r : \mathtt{T} \arrow[d] & & q : \mathtt{F} \arrow[d] & \\
p : \mathtt{F} & \neg q : \mathtt{F} \arrow[d] & r : \mathtt{F} & & r : \mathtt{T} \arrow[dl] \arrow[dr] & \\
\times & q : \mathtt{T} & \times & p \wedge r : \mathtt{F} \arrow[d] \arrow[dr] & & q : \mathtt{T} \\
 & \times & & p : \mathtt{F} & r : \mathtt{F} & \times \\
 & & & \times & \times &
\end{tikzcd}$$

The tableau closes, hence the formula is a tautology.

1.5.1 Exercises

For each of the following exercises use Semantic Tableaux.

1. Check which of the following formulae are tautologies.

   a) $((p \to q) \to q) \to q$

   b) $((q \to p) \to q) \to q$

   c) $((p \to q) \wedge (p \to \neg q)) \to \neg p$

   d) $((p \vee q) \to \neg r) \to \neg(\neg q \wedge r)$

   e) $((p \to q) \wedge (p \to r)) \to (p \to (q \wedge r))$

   f) $((p \to r) \wedge (q \to r)) \to ((p \wedge q) \to r)$

   g) $((p \to r) \vee (q \to r)) \to ((p \vee q) \to r)$

   h) $((p \to r) \wedge (q \to r)) \to ((p \vee q) \to r)$

   i) $p \to ((q \to r) \to ((p \to q) \to r))$

   j) $(p \to (q \to r)) \to ((p \to q) \to (p \to r))$

2. Check which of the following logical consequences are valid.

   a) $\neg p \to q, \neg p \to \neg q \models p$

b) $p \to (q \wedge r) \models (p \to q) \wedge (p \to r)$

c) $(p \wedge q) \to r \models (p \to r) \wedge (q \to r)$

d) $(p \wedge q) \to r \models (p \to r) \vee (q \to r)$

e) $(p \vee q) \to r \models (p \to r) \wedge (q \to r)$

f) $(\neg p \wedge q) \to \neg r, r \models p \vee \neg q$

3. Let P, Q, R, and S be propositions. Check whether:

   a) if $P \to Q$, $Q \to R$ and $P \wedge S$ are true, then $R \wedge S$ is true;

   b) if R and $(P \wedge \neg R)$ are false, and $\neg P \to \neg Q$ is true, then Q is false;

   c) if $Q \to (R \wedge S)$ is true and $Q \wedge S$ is false, then $R \wedge Q$ is false.

   d) if $P \to Q$ and $Q \to (R \vee S)$ are true and $P \to R$ is false, then S is true;

   e) if $Q \to (R \wedge S)$ is true and $Q \wedge S$ is false, then Q is false.

4. Check the validity of each of following inference rules.

   a)
$$\frac{(p \vee q) \to r, \ p \vee r}{r}$$

   b)
$$\frac{p \to q, \ \neg q \vee r, \ \neg r}{\neg p}$$

   c)
$$\frac{p \to q, \ p \vee \neg r, \ \neg r}{q \to r}$$

   d)
$$\frac{\neg p \to \neg q, \ \neg(p \wedge \neg r), \ \neg r}{\neg q}$$

   e)
$$\frac{p \to q, \ r \vee (s \wedge \neg q), \ \neg r}{\neg p}$$

   f)
$$\frac{p \to q, \ q \to (r \vee s), \ \neg(p \to r)}{s}$$

5. Check the logical correctness of each of the following propositional arguments.

   a)

   If the triangle ABC has a right angle, then it is not equilateral.
   The triangle ABC does not have a right angle.

   Therefore, the triangle ABC is equilateral.

b) In the following argument n is a certain integer.

If n is divisible by 2 and n is divisible by 3, then n is divisible by 6.
If n is divisible by 6, then n is divisible by 2.
n is not divisible by 3.

Therefore, n is not divisible by 6.

c)

If Thomas likes logic, then he is clever.
If Thomas is clever, then he is rich.

Therefore, if Thomas likes logic, then he is rich.

d)

If Christina is not a good student, then she is not clever or she is lazy.
If Christina is clever, then she is not lazy.

Therefore, if Christina is clever, then she is a good student.

e)

If Victor studies hard and he is clever, then he will go to a university.
If Victor does not study hard, then he will not get a good job.

Therefore, if Victor is clever, then he will go to a university or will not get a good job.

1.6 Axiomatic Systems in Propositional Logic

1.6.1 Description

Axiomatic systems are the oldest and simplest to describe (but not to use!) type of deductive systems. The probably first prototype of an axiomatic system can be found in Euclid's *Elements* which present a systematic development of elementary geometry, based on several simple assumptions about points and lines (such as "Every two different points determine exactly one line", "For every line there is a point not belonging to that line", etc.). Using these and (informal) logical reasoning, other geometric facts are derived, and thus the entire body of Euclidean geometry is eventually built. But the *logical* concept of axiomatic system was introduced and formalized only at the beginning of the 20th century by the German mathematician David Hilbert (who also essentially re-wrote much of Euclid's *Elements* in a modern and rigorous style) so that they are also known as *Hilbert-style systems.*

Axiomatic systems use very few and simple rules of inference, but are based on *axioms.* Here we will build a hierarchy of axiomatic systems for the propositional logic, by gradually adding axioms for the logical connectives.

If we assume that the only logical connectives are $\to$ and $\neg$ and all others are definable in terms of them, then the axiomatic system comprises the following axioms and rules:

Axioms schemes:

($\to$ 1) $A \to (B \to A)$;

($\to$ 2) $(A \to (B \to C)) \to ((A \to B) \to (A \to C))$;

($\to$ 3) $(\neg B \to \neg A) \to ((\neg B \to A) \to B)$.

The only *rule of inference* is **Modus ponens:**

$$\frac{A, A \to B}{B}.$$

We will denote this system by $\mathbf{H}(\to, \neg)$.

If we consider the conjunction as a primitive, too, rather than definable connective, then it suffices to add the following axiom schemes for it:

($\wedge$1) $(A \wedge B) \to A$;

($\wedge$2) $(A \wedge B) \to B$;

($\wedge$3) $(A \to B) \to ((A \to C) \to (A \to B \wedge C))$.

Likewise, the following axiom schemes suffice for the disjunction:

($\vee$1) $A \to A \vee B$;

($\vee$2) $B \to A \vee B$;

($\vee$3) $(A \to C) \to ((B \to C) \to (A \vee B \to C))$.

The axiomatic system comprising all instances of these axiom schemes and the rule Modus Ponens will be called simply $\mathbf{H}$.

1.6.2 Some derivations

A formula A which can be derived by using the axioms and applying successively Modus Ponens is said to be *derivable in* **H**, or a *theorem of* **H**, which we will denote by $\vdash_{\mathbf{H}} A$.

Using **H** one can derive logical consequences, too. In order to derive "If $A_1, \dots, A_n$ then B" we add the premises $A_1, \dots, A_n$ to the set of axioms and try to derive B. If we succeed, we say that B is *derivable in **H** from the assumptions* $A_1, \dots, A_n$, denoted $A_1, \dots, A_n \vdash_{\mathbf{H}} B$. In particular, $\varnothing \vdash_{\mathbf{H}} B$ means simply $\vdash_{\mathbf{H}} B$.

Example 17 $\vdash_{\mathbf{H}} (p \wedge (p \to q)) \to q$*:*

1. $\vdash_{\mathbf{H}} (p \wedge (p \to q)) \to p$, *by Axiom* $(\wedge 1)$;
2. $\vdash_{\mathbf{H}} (p \wedge (p \to q)) \to (p \to q)$, *by Axiom* $(\wedge 2)$;
3. $\vdash_{\mathbf{H}} ((p \wedge (p \to q)) \to (p \to q)) \to (((p \wedge (p \to q)) \to p) \to ((p \wedge (p \to q)) \to q))$, *by Axiom* $(\to 2)$;
4. $\vdash_{\mathbf{H}} (p \wedge (p \to q)) \to p) \to ((p \wedge (p \to q)) \to q)$, *by 2,3 and Modus Ponens;*
5. $\vdash_{\mathbf{H}} (p \wedge (p \to q)) \to q$, *by 1,4 and Modus Ponens.*

One can check that all axioms of **H** are tautologies and therefore, since the rule Modus Ponens is valid, **H** can only derive tautologies when using these axioms as premises. By the same argument, **H** can only derive *valid* logical consequences. Therefore, the axiomatic system **H** is *sound*, or *correct.*

In fact, it can be proved that **H** can derive all valid logical consequences, and in particular, all tautologies, i.e., it is *complete.* Thus, **H** captures precisely the notion of propositional logical consequence.

However, this axiomatic system is not very suitable for practical derivations as is, because even simple cases may require involved derivations. For instance, as a challenge, try deriving the tautology $p \to p$. Still, the derivations in **H** can be simplified significantly by using the following result which allows us to introduce, and later eliminate auxiliary assumptions in the derivations:

Theorem 18 *(Deduction Theorem) For any formulae A, B, and a set of formulae* Γ*:*

$$\Gamma \cup \{A\} \vdash_{\mathbf{H}} B \text{ iff } \Gamma \vdash_{\mathbf{H}} A \to B.$$

While one direction of this theorem (which?) is an immediate application of Modus Ponens, the proof of other requires a more involved argument which will not be presented here (but see the exercises).

Using the Deduction Theorem the derivation of $\vdash_{\mathbf{H}} p \to p$ becomes trivial. The derivation in the example above can be simplified, too:

1. $p \wedge (p \to q) \vdash_{\mathbf{H}} p$, by Axiom $(\wedge 1)$ and the Deduction Theorem;
2. $p \wedge (p \to q) \vdash_{\mathbf{H}} p \to q$ by Axiom $(\wedge 2)$ and the Deduction Theorem;
3. $p \wedge (p \to q) \vdash_{\mathbf{H}} q$ by 1,2, and Modus Ponens;
4. $\vdash_{\mathbf{H}} (p \wedge (p \to q)) \to q$ by 3 and the Deduction Theorem.

Here is one more example of a derivation in **H** using the Deduction Theorem.

Example 19 $A, \neg A \vdash_{\mathbf{H}} B$:

1. $\neg A \vdash_{\mathbf{H}} \neg B \to \neg A$, *by Axiom* $(\to 1)$ *and the Deduction Theorem;*
2. $\neg A \vdash_{\mathbf{H}} (\neg B \to A) \to B)$, *by 1, Axiom* $(\to 3)$ *and Modus Ponens;*
3. $A \vdash_{\mathbf{H}} \neg B \to A$ *by Axiom* $(\to 1)$ *and the Deduction Theorem;*
4. $A, \neg A \vdash_{\mathbf{H}} B$, *by 2,3 and Modus Ponens.*

Note that adding more premises does not affect the validity of derivations, but can only possibly add more derivable formulae. This justifies step 4 above.

Example 20 $p, \neg p \vdash_{\mathbf{H}} q$*:*

1. $\neg p \vdash_{\mathbf{H}} \neg q \to \neg p$, *by Axiom* $(\to 1)$ *and the Deduction Theorem;*
2. $\neg p \vdash_{\mathbf{H}} (\neg q \to p) \to q$, *by 1, Axiom* $(\to 3)$*, and Modus Ponens;*
3. $p \vdash_{\mathbf{H}} \neg q \to p$, *by Axiom* $(\to 1)$ *and the Deduction Theorem;*
4. $p, \neg p \vdash_{\mathbf{H}} q$, *by 2,3, and Modus Ponens.*

1.6.3 Exercises

1. Derive the following in the axiomatic system **H,** using the Deduction Theorem. (Hint: for some of these exercises use the previous ones.)

   a) $(A \wedge B) \to C \vdash_{\mathbf{H}} A \to (B \to C)$.

   b) $A \to (B \to C) \vdash_{\mathbf{H}} (A \wedge B) \to C$.

   c) $\vdash_{\mathbf{H}} A \to ((B \to C) \to ((A \to B) \to (A \to C)))$.

   d) $\vdash_{\mathbf{H}} ((A \to B) \wedge (B \to C)) \to (A \to C)$

   e) $((A \to B) \wedge (A \to C)) \to (A \to (B \wedge C))$

   f) $(A \to B), (A \to C) \vdash_{\mathbf{H}} A \to (B \wedge C)$

   g) $\vdash_{\mathbf{H}} ((A \to C) \wedge (B \to C)) \to ((A \vee B) \to C)$

   h) If $\neg A \vdash_{\mathbf{H}} \neg B$ then $B \vdash_{\mathbf{H}} A$.

   i) If $\neg A, B \vdash_{\mathbf{H}} \neg B$ then $B \vdash_{\mathbf{H}} A$.

   j) $\neg\neg A \vdash_{\mathbf{H}} A$.

   k) $A \vdash_{\mathbf{H}} \neg\neg A$.

   l) If $B \vdash_{\mathbf{H}} A$ then $\neg A \vdash_{\mathbf{H}} \neg B$.

   m) $(B \to A) \to ((B \to \neg A) \to \neg B)$.

   n) If $A, B \vdash_{\mathbf{H}} C$ and $A, \neg B \vdash_{\mathbf{H}} C$ then $A \vdash_{\mathbf{H}} C$.

   o) If $\neg A, B \vdash_{\mathbf{H}} A$ then $\neg A \vdash_{\mathbf{H}} \neg B$.

   p) If $A \vdash_{\mathbf{H}} \neg B$ then $A, B \vdash_{\mathbf{H}} C$.

q) $\vdash_{\mathbf{H}} A \vee \neg A$

r) $\vdash_{\mathbf{H}} \neg(A \wedge \neg A)$

2. Prove that every theorem of **H** is a tautology.

1.7 Natural deduction

1.7.1 Description

One of the practically most convenient deductive systems is called **Natural Deduction** (**ND**). It is based on a set of inference rules which naturally reflect the logical meaning of the propositional connectives and provide a well-structured way of formal reasoning, which closely resembles a good and correct informal argumentation.

ND has no axioms, but several inference rules (listed further) including a pair of rules for each logical connective: an **introduction rule**, which produces a conclusion containing that connective as the main one, and an **elimination rule**, in which the connective occurs as the main connective of a premise. To reduce the number of rules, $\leftrightarrow$ is assumed to be defined in terms of $\rightarrow$ and $\wedge$. Note that, since $\neg A \equiv A \rightarrow \bot$, the rules for $\neg$ can be regarded as particular cases of the corresponding rules for $\rightarrow$. Also, there are two additional rules: $(\bot)$ and (RA) which will be discussed further.

The derivation in **ND** consists of successive application of the inference rules, using as premisses the initial assumptions or already derived formulae, as well as *additional assumptions* which can be added at any step of the derivation. Some of the rules allow for **cancelation** (or, **discharge**) **of assumptions**, which is indicated by putting them in square brackets. The idea of the additional assumptions is that they only play an auxiliary role in the derivation, and when not needed anymore they are cancelled, but *only* at an application of an appropriate rule which allows such cancelation. Note that the cancelation of an assumption, when the rule allows it, is a *right, but not an obligation,* so an assumption can be re-used several times before being cancelled. However, all assumptions which have not been cancelled during the derivation *must be declared* in the list of assumption from which the conclusion is proved to be a logical consequence. Therefore, if we want to prove that a formula C is a logical consequence from a set of assumptions Γ, then any assumption which is not in Γ must be cancelled during the derivation.

Before formally introducing the rules, let us discuss their justification.

$(\wedge I)$ To prove the truth of a conjunction $A \wedge B$, we have to prove the truth of each of A and B.

$(\wedge E)$ The truth of a conjunction $A \wedge B$, implies the truth of each of the conjuncts.

$(\vee I)$ The truth of a disjunction $A \vee B$ follows from the truth of either disjunct.

$(\vee E)$ If the premise is a disjunction $A \vee B$, we reason *per cases*, i.e. we consider separately each of the two possible cases for that disjunction to be true: *Case 1:* A is true, and *Case 2:* B is true. If we succeed to prove that in each of these cases the conclusion C follows, then we have a proof that C follows from $A \vee B$.

$(\rightarrow I)$ To prove the truth of an implication $A \rightarrow B$, we assume that (besides all premises) the antecedent A is true and try to prove that the consequent B is true.

$(\rightarrow E)$ This is the Detachment rule (Modus Ponens) which we have already discussed.

$(\neg I)$ To prove a negation $\neg A$ we can apply *proof by contradiction*, viz. assume A (which is equivalent to the negation of $\neg A$) and try to reach a contradiction.

$(\neg E)$ The falsum follows from any contradiction.

($\bot$) "*Ex falso sequitur quodlibet*": from a false assumption anything can be derived.

(RA) This rule is called '**Reductio ad absurdum**'. It formalizes the method of *proof by contradiction,* or *proof from the contrary:* if the assumption that A is false (i.e. $\neg A$ is true) leads to a contradiction, then A must be true.

Rules for Natural Deduction in propositional logic

(The vertical dots below indicate derivations.)

Introduction rules: **Elimination rules:**

$$(\wedge I)\quad \frac{A, B}{A \wedge B} \qquad\qquad (\wedge E)\quad \frac{A \wedge B}{A}, \quad \frac{A \wedge B}{B}$$

$$(\vee I)\quad \frac{A}{A \vee B}, \quad \frac{B}{A \vee B} \qquad\qquad (\vee E)\quad \frac{A \vee B \quad \begin{array}{c}[A]\\ \vdots\\ C\end{array} \quad \begin{array}{c}[B]\\ \vdots\\ C\end{array}}{C}$$

$$(\rightarrow I)\quad \frac{\begin{array}{c}[A]\\ \vdots\\ B\end{array}}{A \rightarrow B} \qquad\qquad (\rightarrow E)\quad \frac{A,\ A \rightarrow B}{B}$$

$$(\neg I)\quad \frac{\begin{array}{c}[A]\\ \vdots\\ \bot\end{array}}{\neg A} \qquad\qquad (\neg E)\quad \frac{A,\ \neg A}{\bot}$$

$$(\bot)\quad \frac{\bot}{A} \qquad\qquad (RA)\quad \frac{\begin{array}{c}[\neg A]\\ \vdots\\ \bot\end{array}}{A}$$

Exercise 21 *Prove that all rules of* ***ND*** *are valid.*

Exercise 22 *Find valid introduction and elimination rules for the biconditional* $\leftrightarrow$.

It can be proved that **ND** is sound and complete, and therefore it captures precisely the notion of propositional logical consequence, too.

1.7.2 Some examples of derivations in Natural Deduction

1. Commutativity of the conjunction: $A \wedge B \vdash_{\mathbf{ND}} B \wedge A$

$$(\wedge I)\dfrac{(\wedge E)\dfrac{A\wedge B}{B}\qquad (\wedge E)\dfrac{A\wedge B}{A}}{B\wedge A}$$

Note that an assumption $A \wedge B$ was used twice.

2. $\vdash_{\mathbf{ND}} A \to \neg\neg A$:

$$(\to I)\dfrac{(\neg I)\dfrac{(\neg E)\dfrac{[A]^2,\ [\neg A]^1}{\bot}}{\neg\neg A}\,1}{A \to \neg\neg A}\,2$$

In this derivation two additional assumptions have been made: A and $\neg A$. The label 1 indicates the application of the rule which allows the cancelation of the assumption $\neg A$, while label 2 indicates the cancelation of A.

3. $\vdash_{\mathbf{ND}} \neg\neg A \to A$:

$$(\to I)\dfrac{(RA)\dfrac{(\neg E)\dfrac{[\neg\neg A]^2,\ [\neg A]^1}{\bot}}{A}\,1}{\neg\neg A \to A}\,2$$

Note the application of (RA) in this derivation. It can be proved that the derivation cannot be done without this, or an equivalent rule.

4. $\vdash_{\mathbf{ND}} (A \to (B \to C)) \to ((A \wedge B) \to C)$:

$$(\to I)\dfrac{(\to I)\dfrac{(\to E)\dfrac{(\wedge E)\dfrac{[A\wedge B]^1}{B}\qquad (\to E)\dfrac{(\wedge E)\dfrac{[A\wedge B]^1}{A},\qquad [A\to(B\to C)]^2}{B\to C}}{C}}{(A\wedge B)\to C}\,1}{(A\to(B\to C))\to((A\wedge B)\to C)}\,2$$

From now on we will usually omit the rule labels of the steps in the derivations but the reader should be able to figure out which rule is applied at each step.

5. $A \to B \vdash_{\mathbf{ND}} \neg B \to \neg A$:

$$\dfrac{\dfrac{\dfrac{\dfrac{[A]^1,\ A\to B}{B},\ [\neg B]^2}{\bot}}{\neg A}\,1}{\neg B \to \neg A}\,2$$

6. $\neg B \to \neg A \vdash_{\mathbf{ND}} A \to B$:

$$(\to I)\,\dfrac{\dfrac{\dfrac{\dfrac{[\neg B]^1,\ \neg B \to \neg A}{[\neg A]},\ [A]^2}{\bot}}{B}\,1}{A \to B}\,2$$

   Again, this derivation requires the use of (RA).

7. $A \vee B \vdash_{\mathbf{ND}} \neg A \to B$:

$$\dfrac{A \vee B \qquad \dfrac{\dfrac{\dfrac{[\neg A]^1, [A]^3}{\bot}}{B}}{\neg A \to B}\,1 \qquad \dfrac{[\neg A]^2, [B]^3}{\neg A \to B}\,2}{\neg A \to B}\,3$$

Note the application of the rule $(\vee E)$ (reasoning per cases) in the last step of this derivation.

1.7.3 Exercises

1. Show that the following tautologies are theorems of **ND**:

   a) $(p \wedge (p \to q)) \to q$

   b) $p \to \neg\neg p$

   c) $\neg\neg p \to p$ (Hint: you will have to use (RA).)

   d) $p \vee \neg p$ (Hint: likewise.)

   e) $((p \to q) \wedge (p \to \neg q)) \to \neg p$

   f) $((p \to q) \wedge (q \to r)) \to (p \to r)$

   g) $((p \to q) \wedge (p \to r)) \to (p \to (q \wedge r))$

   h) $((p \to r) \wedge (q \to r)) \to ((p \vee q) \to r)$

   i) $p \to ((q \to r) \to ((p \to q) \to (p \to r)))$

2. Show that:

   a) If $A \vdash_{\mathbf{ND}} B$ then $\neg B \vdash_{\mathbf{ND}} \neg A$.

   b) If $\neg B \vdash_{\mathbf{ND}} \neg A$ then $A \vdash_{\mathbf{ND}} B$.

3. Derive the following:

   a) $A \to C, B \to C \vdash_{\mathbf{ND}} (A \vee B) \to C$

   b) $A \to \neg A \vdash_{\mathbf{ND}} \neg A$

c) $(A \lor B) \to C \vdash_{\mathbf{ND}} (A \to C) \land (B \to C)$

d) $\neg A \to B, \neg A \to \neg B \vdash_{\mathbf{ND}} A$

e) $(\neg A \land B) \to \neg C, C \vdash_{\mathbf{ND}} A \lor \neg B$

f) $\neg A \to B \vdash_{\mathbf{ND}} A \lor B$

g) $\neg(A \land B) \vdash_{\mathbf{ND}} \neg A \lor \neg B$

h) $\neg A \lor \neg B \vdash_{\mathbf{ND}} \neg(A \land B)$

i) $\neg(A \lor B) \vdash_{\mathbf{ND}} \neg A \land \neg B$

j) $\neg A \land \neg B \vdash_{\mathbf{ND}} \neg(A \lor B)$

k) $\neg(A \to B) \vdash_{\mathbf{ND}} A \land \neg B$

l) $A \land \neg B \vdash_{\mathbf{ND}} \neg(A \to B)$.

1.8 Normal forms. Propositional resolution.

The deductive systems presented so far are, more or less, suitable for human use but not so much (except for semantic tableau) for a computer-based execution. Here we will present yet another deductive system which is particularly suitable for automation, as it is based on no axioms and a single rule of inference, called *(Propositional) Resolution.* Just like the previously studied deductive systems, it is sound and complete for Propositional Logic.

The Resolution rule is very simple:

$$\textbf{RES} \qquad \frac{A \vee C, \quad B \vee \neg C}{A \vee B},$$

where A, B, C are any propositional formulae. The formula C is called a **resolvent** of $A \vee C$ and $B \vee \neg C$, and we write $A \vee C = Res(A \vee C, B \vee \neg C)$.

Theorem 23 *The rule of Propositional Resolution is valid.*

Proof. Exercise. ■

Theorem 24 *The rule of Propositional Resolution preserves satisfiability: if the two premises are simultaneously satisfiable, then the conclusion is satisfiable, too.*

Proof. Exercise. ■

1.8.1 Conjunctive and disjunctive normal forms of propositional formulae

The Resolution rule is particularly efficient when applied to formulae pre-processed in so called *clausal form*, which can be obtained immediately from *conjunctive normal form.* Normal forms are useful for other technical purposes, too, and we will devote this subsection to them.

Definition 25

1. *A* ***literal*** *is a propositional variable or its negation.*
2. *An* ***elementary disjunction*** *(respectively,* ***elementary conjunction)*** *is a disjunction (respectively, conjunction) of literals.*
3. *A* ***disjunctive normal form (DNF)*** *is a disjunction of elementary conjunctions.*
4. *A* ***conjunctive normal form (CNF)*** *is a conjunction of elementary disjunctions.*

Examples:

- p, $\neg q$, $p \vee \neg q$, $p \vee \neg p \vee q \vee \neg r$ are elementary disjunctions;
- p, $\neg q$, $\neg p \wedge q$, $\neg p \wedge q \wedge \neg r \wedge \neg p$ are elementary conjunctions.
- p, $\neg q$, $p \wedge \neg q$, $p \vee \neg q$, $(p \wedge \neg p) \vee \neg q$, $(r \wedge q \wedge \neg p) \vee (\neg q \wedge p) \vee (\neg r \wedge p)$ are disjunctive normal forms;
- p, $\neg q$, $p \wedge \neg q$, $p \vee \neg q$, $p \wedge (\neg p \vee \neg q)$, $(r \vee q \vee \neg r) \wedge \neg q \wedge (\neg p \vee r)$ are conjunctive normal forms.

The normal forms are convenient for various symbolic manipulations and logical computations. Also, they are used for construction and optimization of logical circuits in computer systems design. That is why the following fact is quite important.

Theorem 26 *Every propositional formula is equivalent to a disjunctive normal form and to a conjunctive normal form.*

We shall give two methods for construction of such equivalent normal forms.

I. *The first method* is based on the following algorithm which transforms any formula into a DNF, respectively CNF, using some of the logical equivalences listed above.

1. Eliminate all occurrences of $\leftrightarrow$ and $\rightarrow$ using the logical equivalences
$$A \rightarrow B \equiv \neg A \vee B,$$
$$A \leftrightarrow B \equiv (A \rightarrow B) \wedge (B \rightarrow A).$$
2. Import all negations in front of the propositional variables, using the logical equivalences listed above.
3. For a DNF: distribute all conjunctions over disjunctions using $p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r)$.
4. Respectively, for a CNF: distribute all disjunctions over conjunctions using $p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$.

Throughout this process the formulae can be simplified by using commutativity, associativity, and idempotency of $\vee$ and, as well as:

$p \vee \neg p \equiv \top$; $p \wedge \neg p \equiv \bot$;

$p \wedge \top \equiv p$; $p \wedge \bot \equiv \bot$;

$p \vee \top \equiv \top$; $p \vee \bot \equiv p$;

Example 27 $(p \wedge \neg r) \rightarrow (p \leftrightarrow \neg q)$

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

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

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

For a DNF we further distribute $\wedge$ over $\vee$ and simplify:

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

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

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

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

For a CNF we distribute $\vee$ over $\wedge$ and simplify:

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

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

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

$\equiv \neg p \vee r \vee \neg q.$

As we see, in this case the CNF turns out to be a DNF as well, even simpler than the one we obtained above. The problem of *minimization* of normal forms, which is of practical importance, will not be discussed here.

The second method constructs the normal forms directly from the truth-table of the given formula. We shall outline it for a DNF.

Given the truth-table of the formula A we consider all rows (i.e. all assignments of truth values to the occurring variables) where the truth value of A is `true`. If there are no such rows, the formula is a contradiction and a DNF for it is e.g. $p \wedge \neg p$. Otherwise, with every such row we associate an elementary conjunction in which all variables assigned value `T` occur positively, while those assigned value `F` occur negated. For instance, the assignment `F`, `T`, `F` to the variables p, q, r is associated with the elementary conjunction $\neg p \wedge q \wedge \neg r$. Note that such an elementary conjunction is true only for the assignment with which it is associated.

Exercise 28 *Show that the disjunction of all elementary conjunctions associated with the rows in the truth-table of a formula A is logically equivalent to A.*

Example 29 *The formula $p \leftrightarrow \neg q$ has a truth table*

p	q	$p \leftrightarrow \neg q$
T	T	F
T	F	T
F	T	T
F	F	F

The corresponding DNF is $(p \wedge \neg q) \vee (\neg p \wedge q)$. Check this!

Exercise 30 *Outline a similar method for construction of a CNF from the truth-table of a formula. Prove your claim.*

1.8.2 Clausal resolution

Definition 31

1. *A* ***clause*** *is essentially an elementary disjunction $l_1 \vee \ldots \vee l_n$ but written as a set of literals $\{l_1, \ldots, l_n\}$.*
2. *The* ***empty clause*** *$\{\}$ is a clause containing no literals; a* ***unit clause*** *is a clause containing only one literal.*
3. *A* ***clausal form*** *is a (possibly empty) set of clauses, written as a list: $C_1 \ldots C_k$. It represents the* conjunction *of these clauses.*

Thus, every CNF can be re-written in a clausal form, and therefore every propositional formula is equivalent to one in a clausal form.

Example: the clausal form of the CNF-formula $(p\vee\neg q\vee\neg r)\wedge\neg p\wedge(\neg q\vee r)$ is $\{p,\neg q,\neg r\}\{\neg p\}\{\neg q,r\}$.

The Resolution rule can be rewritten for clauses as follows:

$$\mathbf{CL-RES} \qquad \frac{\{A_1,\dots,C,\dots,A_m\}\{B_1,\dots,\neg C,\dots,B_n\}}{\{A_1,\dots,A_m,B_1,\dots,B_n\}}.$$

The clause $\{A_1,\dots,A_m,B_1,\dots,B_n\}$ is a **resolvent** of the clauses $\{A_1,\dots,C,\dots,A_m\}$ and $\{B_1,\dots,\neg C,\dots,B_n\}$.

Example 32

$$\frac{\{p,q,\neg r\}\{\neg q,\neg r\}}{\{p,\neg r,\neg r\}},$$

$$\frac{\{\neg p,q,\neg r\}\{r\}}{\{\neg p,q\}},$$

$$\frac{\{\neg p\}\{p\}}{\{\}}.$$

Note that two clauses can have more than one resolvents, e.g.:

$$\frac{\{p,\neg q\}\{\neg p,q\}}{\{p,\neg p\}},\ \frac{\{p,\neg q\}\{\neg p,q\}}{\{\neg q,q\}}.$$

However, it is wrong to apply the Resolution rule for both pairs of complementary literals concurrently and obtain

$$\frac{\{p,\neg q\}\{\neg p,q\}}{\{\}}.$$

Also, note that sometimes the resolvent can – and should – be simplified, by removing duplicated literals on the fly:

$$\{A_1,\dots,C,C,\dots,A_m\} \ \Rightarrow\ \{A_1,\dots,C,\dots,A_m\}.$$

For instance:

$$\frac{\{p,\neg q,\neg r\}\{q,\neg r\}}{\{p,\neg r\}} \text{ instead of } \frac{\{p,\neg q,\neg r\}\{q,\neg r\}}{\{p,\neg r,\neg r\}}.$$

1.8.3 Resolution-based derivations

The method of Resolution works similarly to Semantic tableau. In order to determine whether a logical consequence $A_1,\dots,A_n \models B$ holds using the method of Resolution, we negate the conclusion B and transform each of the formulae $A_1,\dots,A_n,\neg B$ into clausal form. Then we test whether the resulting set of clauses is unsatisfiable, by looking for a resolution-based proof of the empty clause from that set of clauses. Formally:

Definition 33 *A* ***resolution-based derivation*** *of a formula C from a list of formulae $A_1,\dots,A_n$ is a derivation of the empty clause $\{\}$ from the set of clauses obtained from $A_1,\dots,A_n,\neg C$, by successive applications of the rule of Propositional Resolution.*

Example 34 *Prove* $p \to q, q \to r, \models p \to r$.

First, transform $p \to q, q \to r, \neg(p \to r)$ *to clausal form:*

$$C_1 = \{\neg p, q\}, C_2 = \{\neg q, r\}, C_3 = \{p\}, C_4 = \{\neg r\}.$$

Now, applying Resolution successively:

$C_5 = \{q\} = Res(C_1, C_3)$;

$C_6 = \{r\} = Res(C_2, C_5)$;

$C_6 = \{\} = Res(C_4, C_6)$.

The derivation of the empty clause completes the proof.

For more details and examples on propositional resolution, see links to supplementary readings on the course website.

1.8.4 Exercises

1. Prove that the rule of Propositional Resolution is valid.
2. Show that the disjunction of all elementary conjunctions associated with the rows in the truth-table of a formula A is logically equivalent to A.
3. Outline a similar method for construction of a CNF from the truth-table of a formula as in Example **??**. Prove your claim.
4. Construct a DNF and a CNF, equivalent to each of the following formulae, using both methods.

   a) $\neg(p \leftrightarrow q)$

   b) $((p \to q) \wedge \neg q) \to p$

   c) $(p \leftrightarrow \neg q) \leftrightarrow r$

   d) $p \to (\neg q \leftrightarrow r)$

   e) $(\neg p \wedge (\neg q \leftrightarrow p)) \to ((q \wedge \neg p) \vee p)$

5. Using the method of Resolution check which of the following formulae are tautologies.

   a) $((p \to q) \to q) \to q$

   b) $((q \to p) \to q) \to q$

   c) $((p \to q) \wedge (p \to \neg q)) \to \neg p$

   d) $((p \vee q) \to \neg r) \to \neg(\neg q \wedge r)$

   e) $((p \to q) \wedge (p \to r)) \to (p \to (q \wedge r))$

   f) $((p \to r) \wedge (q \to r)) \to ((p \wedge q) \to r)$

   g) $((p \to r) \vee (q \to r)) \to ((p \vee q) \to r)$

h) $((p \to r) \land (q \to r)) \to ((p \lor q) \to r)$

i) $p \to ((q \to r) \to ((p \to q) \to r))$

j) $(p \to (q \to r)) \to ((p \to q) \to (p \to r))$

6. Using the method of Resolution check the following logical consequences:

a) $\neg p \to q, \neg p \to \neg q \models p$

b) $p \to r, q \to r \models (p \lor q) \to r$

c) $(p \land q) \to r \models (q \to r) \lor (p \to r)$

d) $p \to q, p \lor \neg r, \neg r \models \neg q \lor r$

e) $(p \lor q) \to r \models (p \to r) \land (q \to r)$

f) $(\neg p \land q) \to \neg r, r \models p \lor \neg q$

g) $p \to q, r \lor (s \land \neg q), \models \neg r \to \neg p$

h) $p \to q, q \to (r \lor s), \neg(p \to r) \models s$

2 Understanding first-order logic

Propositional logic can formalize some of our reasoning, but it cannot grasp the logical structure, nor the truth behaviour of very simple sentences such as

"$x + 2$ is greater than 5.",

"There exists y such that $y^2 = 2$.",

"For every real number x, if x is greater than 0, then there exists a real number y such that y is less than 0 and y^2 equals x . ",

or, for a non-mathematical example, take

"Every man loves a woman".

Indeed, note that an expression such as "$x + 2$ is greater than 5" is *not* a proposition, for it can be true or false depending on the choice of x. Neither is "There exists y such that $y^2 = 2$" a proposition until the range of possible values of y is specified: if y is an integer, then the statement is false, but if y can be any real number, then it is true. As for the third sentence above, it *is* a proposition, but its truth heavily depends on its internal logical structure and mathematical meaning of all phrases involved, and those are not tractable on propositional level.

All these sentences take us out of the simple world of propositional logic into the realm of **first-order logic**[1] (also known as **predicate logic**, or just **classical logic**), the basic concepts of which we will introduce and discuss here.

First-order logic, just like every logical system, has two major aspects:

- precise **syntax**, involving a formal language, called a **first-order language**, that enables us to express statements in a uniform way, by means of **logical formulae**;
- formal **logical semantics**, specifying the meaning of all components of the language by means of their **interpretation** in a mathematical structure, and formal **truth definitions**, extending the truth-tables for the propositional connectives.[2]

Here we will discuss the basic components of the syntax and semantics of first-order logic and their relevance to mathematical reasoning.

[1]The term 'first-order' will be explained later.

[2]The reader who has some experience with computer programming languages should find the concepts of syntax and semantics familiar.

2.1 Basic concepts of first-order (predicate) logic.

2.1.1 First-order structures

First-order logic is a logical framework for formalizing statements and reasoning about universes represented as so called *first-order structures.* Here we will first discuss the basic components of a first-order structure, and then will give a more formal definition.

- **Domains**. When we reason and make statements about objects, we have in mind a certain *domain of discourse* where these objects live. In mathematics, that domain would usually consist of *numbers* (integers, rationals, reals, etc.), *vectors*, *geometric figures* (points, lines, triangles, spheres, etc.), *sets*, etc. mathematical objects. A domain in a non-mathematical discourse may consist of material objects, human beings, ideas, or just about anything else. In either case, it is important that we have specified our domain of discourse, so that we know what we are talking about. Moreover, as one of the sentences mentioned above suggests, the same statement can be true or false, depending on the domain in which it is considered.

- **Predicates**. When we reason about the objects from our domain of discourse, we usually make statements about properties they have or do not have. For instance, talking about integers, we may discuss properties like being 'positive', 'divisible by 3', 'not greater than 1999', etc. Triangles can be 'obtuse', ' equilateral', etc. A human being can be 'female', 'male', 'young', 'blue-eyed', etc.

  The mathematical term for a property is *predicate.* Predicates need not only concern one object, like all examples mentioned above. Those are called *unary predicates.* We also deal with *binary predicates*, relating two objects, such as: '_ is less than _', '_ is divisible by _' (for numbers); or '_ is a son of _', ' _ loves _' (for humans), etc.; *ternary* predicates, relating three objects, such as '_ is between _and _' (for points on a line), ' _ and _ are the parents of _', (for humans), etc. In general, we can talk about *n-ary* predicates, relating n objects.

  Note that, as long as we specify the meaning (semantics) of a predicate and the objects that it relates, that predicate represents a proposition, i.e. becomes true or false. For example, the propositions '12 is divisible by 6' and '13 is divisible by 7' are both instances of the binary predicate 'x is divisible by y'. Furthermore, we can connect predicates, by using propositional connectives, in compound propositions, just like we did earlier with propositions.

- **Functions**. Typically (but not only) in mathematics we use *functions* to represent operations which, applied to one or several objects, determine an object. Depending on the number of arguments, we talk about *unary, binary,* etc. functions. Standard examples are all arithmetic operations $+, \times$ and more generally, all algebraic functions we have studied at school; and so are 'the mother of _', 'the father of _', etc., in the domain of humans.

  Note that in our logical framework all functions are assumed *total*, i.e. defined for all possible values of the argument or tuple of arguments in the domain. This is quite often not the case, e.g. the subtraction in the domain of natural numbers, division in the domain of reals, or the function "the daughter of _" in the domain of humans. This problem has an easy (albeit artificial) fix, good enough for our formal purposes: we can designate an element u (for *'undefined'*) from the domain and make the function total by assigning u to be its value for all (tuples of) arguments for which it is not defined. For instance, we can extend division by 0 in the domain of reals by putting $x/0 = 17$ for any real x. This may sound reckless, but if proper care is taken when reasoning about division, it will not lead to confusion or contradiction. More on this, later.

- **Constants**[3]. Some objects in the domain can be distinguished in a way that would allow us to make direct references to them, by giving them *names*. Such distinguished objects are called **constants**. Examples in the domain of real numbers are $0, 1, \sqrt{2}, \pi, e$, etc. Note that the notion of a constant pertains to the language of discourse rather than to the domain. For instance, in common calculus we do not have a special name for the least positive solution of the equation $\cos x = x$, but if for some purposes we need to refer directly to it, then we can extend our mathematical language by giving it a name, i.e. make it a constant in our domain.

Now, we are ready to define the general notion of a *structure*.

Definition 35 *A* ***(first-order) structure*** *consists of a non-empty domain, and a family of distinguished functions, predicates, and constants in that domain.*

Here are some examples of structures, that will often be used further:

- $\mathcal{N}$: The set of natural numbers $\mathbf{N}$ with the unary function s (successor, i.e. $s(x) = x+1$), the binary functions $+$ (addition) and $\times$ (multiplication), the predicates $=$, $<$ and $>$, and the constant 0.
- $\mathcal{Z}$: likewise, but with the domain being the set of integers $\mathbf{Z}$, and the additional function $-$ (subtraction).
- With the same functions and predicates we take the domain to be the set of rational numbers $\mathbf{Q}$ or the reals $\mathbf{R}$. The obtained structures will be denoted by $\mathcal{Q}$ and $\mathcal{R}$ respectively.
- $\mathcal{H}$: the domain is the set of all humans, with functions m (for 'the mother of') and f (for 'the father of'), unary predicates M (for 'man') and W (for 'woman'), binary predicates P (for 'parent of'), C (for 'child of'), L (for 'loves'), and constants (names), e.g. John, Mary, Adam, Eve.
- $\mathcal{G}$: the domain is the set of all points and lines in the plane, with unary predicates P for 'point', L for 'line' and the binary predicate I for 'incidence' between a point and a line.
- $\mathcal{S}(U)$: the domain is the family of all elements and subsets of a given 'universe' set U, with binary predicates $=$ for equality and $\in$ for membership.

2.1.2 First-order languages

In order to refer to objects, functions and predicates in our domain, we give them *names*. In mathematics, they are typically symbols or abbreviations, such as $5, \pi, +, =, <, \sqrt{}, \sin$;

[3]Note that the use of the word 'constant' in first-order logic is not exactly the same as in the common mathematical language.)

for humans – the proper names, etc. Together with logical connectives, variables and some auxiliary symbols, these symbolic names determine formal logical languages called *first-order languages.* In particular, with every structure $\mathcal{S}$ we can associate a *first-order language* $\mathcal{L}_{\mathcal{S}}$ for that structure, containing:

1. **Functional, predicate, and constant symbols,** used as names for the functions, predicates and constants we consider in the structure. All these are referred to as **non-logical symbols**.

   We emphasize, that the non-logical symbols are *just names* for functions, predicates and constants, so, in principle, we should use different symbols to denote the former and the latter objects. Later on we will introduce a generic notation that will take care of that, but meanwhile we will afford a little sloppiness and will use the same symbols for functions, predicates and constants in our example structures and for their names in the first-order languages designed for them.

2. **Individual variables**. Quite often, particularly in mathematics, we deal with unknown or unspecified objects (individuals) from the domain of discourse. In order to be able to reason and make statements about such objects, we use *individual variables* to denote them. For instance, talking about numbers, we use phrases like "Take a positive integer n", "For every real number x greater than $\sqrt{2}$...", etc. In natural language instead of variables we usually use pronouns or other syntactic constructions but that often leads to awkwardness or even ambiguity (e.g. " If a man owes money to another man than that man hates the other man"), so the use of variables in mathematics is indispensable and it is very important to learn how to use them properly.

   We also use variables as *placeholders* for the arguments of the various predicates and functions we deal with, and they are much more convenient than the ellipses we have used in the previous paragraphs. Thus, we can talk about the (unary) functions $f(x)$, 'the mother of z' or the (binary) predicates $P(x, y)$ and 'x loves y', etc. .

   We will assume that any first-order language contains an infinite set of individual variables, denoted by VAR. Usually, we will be using letters u, v, w, x, y, z, possibly indexed, to denote individual variables.

3. **Auxiliary symbols**, such as '(', ',' , ')', etc.

4. Last and most important, **logical symbols,** including:

   a) **Propositional connectives**, which we already know, and

   b) **Quantifiers**. Often we use specific phrases to *quantify* over objects of our discourse, such as:[4]

      - "(for) every (objects) x ",

      - "there is (an object) x such that ",

      - "for most (objects) x ",

      - "there are at least 5 (objects) x such that ",

      - "there are more (objects) x than y such that ",

      etc.

[4]Note the use of variables here.

The first two quantifiers are particularly important and many others can be expressed by means of them, so they are given special names and notation:

- The quantifier "for every " is called **universal quantifier**, denoted by $\forall$.
- The quantifier "there exists " is called **existential quantifier**, denoted by $\exists$.

These are the quantifiers used in first-order languages, so they are called **first-order quantifiers**. The term 'first-order' refers to the fact that quantification in these languages is only allowed over individuals in the universe of discourse – called 'first-order objects' – but not over more complex objects such as sets, relations, functions, etc.

Besides the phrases above, the universal quantifier is usually represented by *"all", "for all"* and *"every"* while the existential quantifier can appear as *"there is", "some",* and *"for some"*, particularly in a non-mathematical discourse.

2.1.3 Terms and formulae

Using the symbols in a given first-order language and following certain common syntactic rules we can compose formal expressions which allow us to represent symbolically mathematical statements, to reason about them, and to prove them in a precise, well-structured and logically correct way.

There are two basic syntactic categories in a first-order language: **terms** and **(first-order) formulae**.

Terms

Terms are formal expressions (think of algebraic expressions) built from constant symbols and individual variables, using functional symbols. Terms are used to denote specified or unspecified *individuals,* i.e. elements of the domain.

Here is the formal *inductive* definition of the terms of a first-order language $\mathcal{L}$:

1. *Every constant symbol in $\mathcal{L}$ is a term.*
2. *Every individual variable in $\mathcal{L}$ is a term.*
3. *If $t_1, ..., t_n$ are terms and f is an n-ary functional symbol in $\mathcal{L}$, then $f(t_1, ..., t_n)$ is a term in $\mathcal{L}$.*

Terms that do not contain variables are called **constant terms**.

We denote the set of terms of $\mathcal{L}$ by $TM(\mathcal{L})$.

EXAMPLES:

1. In the language $\mathcal{L}_{\mathcal{N}}$:

   To begin with, 0, $s(0)$, $s(s(0))$, etc. are constant terms. We will denote the term $s(...s(0)...)$, where s occurs n times, by $\mathbf{n}$.[5]

[5]You should distinguish *the term* $\mathbf{n}$, which is a syntactic object, just a string of symbols, from *the number* n which is a mathematical entity.

Hereafter we will be a little informal and will sometimes allow ourselves using infix notation and omitting outermost parentheses in terms, whenever that does not affect the correct reading. With that in mind, here are more examples of terms:

- $+(\mathbf{2}, \mathbf{2})$, which in a more familiar notation is written as $\mathbf{2} + \mathbf{2}$;
- $\times(\mathbf{3}, y)$, written in the usual notation $\mathbf{3} \times y$;
- $(x^2 + x) + \mathbf{5}$, where x^2 is an abbreviation of $x \times x$;
- $x_1 + s((y_2 + \mathbf{3}) \times s(z))$, etc.

2. In the 'human' language $\mathcal{L}_\mathcal{H}$:

   - Mary,
   - x,
   - m(John) (intended meaning: 'the mother of John'),
   - f(m(y)) (intended meaning: 'the father of the mother of y'), etc.

With every term we can associate a construction tree and a parsing tree, just like with propositional formulae. Respectively, for every term t we define the set sub(t) of *subterms* being all terms used in the construction of that term, i.e., all terms with construction trees rooted as subtrees at nodes of the construction tree of t.

Atomic formulae

Applying predicate symbols to terms we can build **atomic formulae.** These are the simplest first-order formulae, with no internal logical structure. They correspond to atomic propositions in propositional logic.

The formal definition:

*If $t_1, ..., t_n$ are terms in a language $\mathcal{L}$ and p is an n-ary predicate symbol in $\mathcal{L}$, then $p(t_1, ..., t_n)$ is an **atomic formula** in $\mathcal{L}$.*

Examples:

1. In $\mathcal{L}_\mathcal{N}$:

   - $< (\mathbf{1}, \mathbf{2})$, or in traditional notation: $\mathbf{1} < \mathbf{2}$;
   - $x = \mathbf{2}$,
   - $\mathbf{5} < (x + \mathbf{4})$,
   - $\mathbf{2} + s(x_1) = s(s(x_2))$,
   - $(x^2 + x) + \mathbf{5} = 0$,
   - $x \times (y + z) = x \times y + x \times z$, etc.

2. In $\mathcal{L}_{\mathcal{H}}$:

   - $x = \mathsf{m}(\mathsf{Mary})$ (intended meaning: 'x is the mother of Mary').
   - $\mathsf{L}(\mathsf{f}(y), y)$ (intended meaning: 'the father of y loves y'), etc.

Formulae

Finally, using atomic formulae and logical connectives we can build compound **formulae**, just like in propositional logic, except that now we can use quantifiers, too.

The inductive definition of formulae of a first-order language $\mathcal{L}$ naturally extends the definition of propositional formulae:

1. *Every atomic formula in $\mathcal{L}$ is a formula in $\mathcal{L}$.*
2. *If A is a formula in $\mathcal{L}$ then $\neg A$ is a formula in $\mathcal{L}$.*
3. *If A, B are formulae in $\mathcal{L}$ then $(A \vee B)$, $(A \wedge B)$, $(A \to B), (A \leftrightarrow B)$ are formulae in $\mathcal{L}$.*
4. *If A is a formula in $\mathcal{L}$ and x is a variable, then $\forall x A$ and $\exists x A$ are formulae in $\mathcal{L}$.*

We denote the set of formulae of $\mathcal{L}$ by $FOR(\mathcal{L})$.

Examples:

1. In $\mathcal{L}_{\mathcal{Z}}$:

   - $(\mathbf{5} < x \wedge x^2 + x - \mathbf{2} = \mathbf{0})$,
   - $\exists x(\mathbf{5} < x \wedge x^2 + x - \mathbf{2} = \mathbf{0})$,
   - $\forall x(\mathbf{5} < x \wedge x^2 + x - \mathbf{2} = \mathbf{0})$,
   - $(\exists y(x = y^2) \to (\neg x < \mathbf{0}))$,
   - $\forall x(\exists y(x = y^2) \to (\neg x < \mathbf{0}))$, etc.

2. In $\mathcal{L}_{\mathcal{H}}$:

   - $\mathsf{John} = \mathsf{f}(\mathsf{Mary}) \to \exists x \mathsf{L}(x, \mathsf{Mary})$;
   - $\exists x \forall z(\neg \mathsf{L}(z, y) \to \mathsf{L}(x, z))$;
   - $\forall x(\exists y(x = \mathsf{m}(y)) \to \exists z(\mathsf{L}(z, x)))$.

As in propositional logic, we will allow ourselves to omit parentheses wherever possible without producing ambiguity. For that purpose we will, again, impose a priority order amongst the logical connectives:

- the unary connectives – negation and quantifiers – have the strongest binding power, i.e. the highest priority,
- then come the conjunction and disjunction,
- then the implication, and
- the biconditional has the lowest priority.

Example:

$$\forall x((\exists y(x = y^2)) \to ((\neg(x < \mathbf{0})) \vee (x = \mathbf{0})))$$

can be simplified to

$$\forall x(\exists y x = y^2 \to \neg x < \mathbf{0} \vee x = \mathbf{0}).$$

On the other hand, sometimes we will be using redundant parentheses in order to improve the readability.

With every first-order formula we can associate a construction tree and a parsing tree, just like with propositional formulae. The only differences are that the leaves of the construction/parsing tree of a formula are labelled with atomic formulae rather than propositional variables, and that internal nodes can also be labelled with pairs $\langle$quantifier, individual variable$\rangle$, such as $\forall x$ or $\exists x$, and these nodes have single successor nodes.

Respectively, for every first-order formula A we define its *main connective*, being the one labeling the root of the construction/parsing tree of the formula, and the set $sub(A)$ of *subformulae* of A being all formulae used in the construction of that formula, i.e., all formulae with construction trees rooted as subtrees at nodes of the construction tree of A.

Unique readability of terms and formulae

Natural languages do not provide a reliable medium for precise reasoning because they are ambiguous: the same phrase or sentence may have several different, yet grammatically correct, readings, and therefore, several different meanings. Eliminating ambiguity is one of the main reasons to use formal logical languages instead. In particular, the first-order languages introduced here are unambiguous in their formal syntax. In particular, it can be proved (though, the proof is long and tedious) that the terms and formulae of any first-order language $\mathcal{L}$ have the *unique readability property*: every term or formula has an essentially unique construction tree, respectively parsing tree, up to the order of listing of the successor nodes. More precisely, for any first-order language $\mathcal{L}$ the following hold:

1. Every occurrence of a functional symbol in a term t from $TM(\mathcal{L})$ is the beginning of a unique subterm of t.
2. Every occurrence of a predicate symbol, $\neg, \exists$, or $\forall$ in a formula A from $FOR(\mathcal{L})$ is the beginning of a unique subformula of A.
3. Every occurrence of any binary connective $\circ \in \{\wedge, \vee, \to, \leftrightarrow\}$ in a formula A from $FOR(\mathcal{L})$ is in a context $(B_1 \circ B_2)$ for a unique pair of subformulae B_1 and B_2 of A.

First-order instances of propositional formulae

Definition 36 *Given a propositional formula A, any uniform substitution of first-order formulae for the propositional variables in A produces a first-order formula, called a **first-order instance of** A.*

For example, substituting $(\mathbf{5} < x)$ for p and $\exists y(x = y^2)$ for q in the propositional formula

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

produces its first-order instance

$$((\mathbf{5} < x) \wedge \neg\exists y(x = y^2)) \rightarrow (\exists y(x = y^2) \vee (\mathbf{5} < x)).$$

Many-sorted first-order structures and languages

Often the domain of discourse involves different *sorts* of objects, e.g., integers and reals; scalars and vectors; points, lines, triangles, circles; etc.

The notions of first-order structures and languages can be extended naturally to *many-sorted* structures and languages, with suitable inter-sort and cross-sort functions and predicates.

However, using several sorts for individuals comes with a technical overhead, e.g., requires different sorts of individual variables, etc. Instead, in this exposition we will use unary predicates to identify the different sorts within a universal domain.

2.1.4 Exercises

1. Define suitable first-order structures and introduce appropriate first-order languages for the following types of mathematical structures: graphs, matrices, rings, fields, vector spaces.
2. Define suitable first-order structures and introduce appropriate first-order languages for the following data types: Booleans, strings, lists, trees.
3. Which of the following are syntactically correct terms in the language $\mathcal{L}_{\mathcal{N}}$? Re-write them in the usual infix notation.
   a) $\times(x, y)$;
   b) $\times(\mathbf{2}, x, y)$;
   c) $+xy$;
   d) $+(\mathbf{5}, \times(x, +(\mathbf{3}, y)))$;
   e) $+(\times(x, +(\mathbf{3}, y)), \times(+(\mathbf{3}, y), x))$;
   f) $\times(s(\times(\mathbf{2}, x), +(\mathbf{3}, y)))$;
   g) $s(\times(\times(\mathbf{2}, x), +(\mathbf{3}, s)))$;
   h) $s(\times(\times(\mathbf{2}, x), +(\mathbf{3}, y)))$;
   i) $\times(s(\times(\mathbf{2}, x)), s(+(s(\mathbf{3}), y)))$;
   j) $s(\times(s(s(x)), s(s(s(\mathbf{2})))), s(\mathbf{2}))$
4. Construct the parsing tree of each of the syntactically correct terms from the previous exercise and determine the set of its subterms.
5. * Prove that every occurrence of a functional symbol in a term t is the beginning of a unique subterm of t.
6. Which of the following are syntactically correct formulae in the language $\mathcal{L}_{\mathcal{N}}$? Assume that terms are written in the usual infix notation and parentheses can be omitted whenever possible or added for better readability.

a) $\neg\forall x(x = \mathbf{0})$;

b) $\forall\neg x(x = \mathbf{0})$;

c) $\forall x(\neg x = \mathbf{0})$;

d) $\forall x(x\neg = \mathbf{0})$;

e) $\forall x(x = \neg\mathbf{0})$;

f) $\neg\forall x(\neg x = \mathbf{0})$;

g) $\neg\exists x(\neg x = 0.5)$;

h) $\forall x(x \vee \neg x)$;

i) $\forall x\forall y((x + y) \rightarrow (y + x))$;

j) $\forall x((x < \mathbf{0}) \rightarrow \forall x(x > \mathbf{0}))$.

k) $\forall x((x + \mathbf{0}) \rightarrow \forall x(x > \mathbf{0}))$.

l) $\forall x(\forall x(x > \mathbf{0}) \rightarrow x < \mathbf{0})$.

m) $\forall y(x < x \rightarrow \forall x(x < x))$;

n) $\forall y((y < y)\neg\forall x(x < x))$;

7. Construct the parsing tree of each of the syntactically correct formulae from the previous exercise and determine its main connective and the set of its subformulae.

8. * Prove that every occurrence of a predicate symbol, $\neg, \exists$, or $\forall$ in a formula A from $FOR(\mathcal{L})$ is the beginning of a unique subformula of A.

9. * Prove that every occurrence of any binary connective $\circ \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$ in a formula A from $FOR(\mathcal{L})$ is in a context $(B_1 \circ B_2)$ for a unique pair of subformulae B_1 and B_2 of A.

2.2 Semantics of first-order logic

Formulae are meant to express statements. However, the meaning of the statement expressed by a formula cannot be determined only from the symbols occurring in the formula and its shape, i.e. from the *syntax.* The meaning of a formula is determined by its *interpretation*, being the first-order structure to which (the statement expressed by) that formula applies, i.e. the domain of discourse and the meaning of the non-logical symbols occurring in the formula that is associated with that structure. In addition, the truth value of the statement expressed by the formula also depends on the *values* assigned to the individual variables in that interpretation. Since formulae belong to first-order languages, we need to define interpretations of entire languages.

2.2.1 Interpretations of first-order languages

An interpretation of a first-order language $\mathcal{L}$ is a *matching* first-order structure $\mathcal{S}$, that is, a structure with a family of distinguished functions, predicates, and constants that correspond to (and match the respective numbers of arguments of) the non-logical symbols in $\mathcal{L}$.

Some first-order languages. like all those that we have considered so far, are designed for specific structures, which are their intended, or *standard interpretations.*

Other first-order languages are designed for *classes of structures.* For instance, the first-order language containing one binary relational symbol R (plus equality) can be regarded as the language of directed graphs, where the intended interpretation of R in any directed graph is the edge relation in that graph. Likewise, the first-order language of (algebraic) groups contains the following non-logical symbols: one binary functional symbol $\circ$, with intended interpretation being the group operation; one unary functional symbol $'$, with intended interpretation being the inverse operation; and one constant symbol e, with intended interpretation the identity element.

Note, however, that every first-order language may have many *unintended* interpretations. For instance, the language for directed graphs can be interpreted in the domain of integers, where with R interpreted as 'divisible by', or in the domain of humans, with R interpreted as 'is a friend of'.

In fact, most of the unintended interpretations are generally meaningless. For instance, the language $\mathcal{L}_{\mathcal{H}}$ can be interpreted in the domain of integers, where e.g., the functional symbol m is interpreted as '$\mathsf{m}(n) = 2n$', f is interpreted as '$\mathsf{f}(n) = n^5 - 1$', the unary predicate M is interpreted as 'is prime' and the unary predicate W is interpreted as 'is greater than 2012', the binary predicate symbols P, C, and L are interpreted respectively as 'is greater than', 'is divisible by', 'has the same remainder modulo 11', and the constant symbols John, Mary, Adam, Eve are interpreted respectively as the numbers $-17, 99, 0$, and 10.

Of course, such unintended interpretations are not interesting, but they must be taken in consideration when judging whether a given first-order formula is *logical valid*, that is, true in *every* possible interpretation. But, that will be discussed later, and now we come back to the meaning of first-order terms and formulae under a given interpretation.

Once a given first-order language $\mathcal{L}$ is interpreted, that is, a matching first-order structure $\mathcal{S}$ is fixed, the *value in* $\mathcal{S}$ of every term t from $TM(\mathcal{L})$ can be 'computed', as soon as all individual variables occurring in t are assigned *values* in $\mathcal{S}$, i.e., elements of $\mathcal{S}$. Then, the meaning of every formula A in $FOR(\mathcal{L})$ can be 'computed', too, just like in propositional logic, from the values of the terms and the interpretation of the predicate symbols occurring in A, and the standard meaning of the logical connectives. The rules for computing this meaning determine the **semantics of first-order logic**. We will spell out these rules without going into more technical details than really necessary.

2.2.2 Variable assignment and term evaluation

As we discussed above, in order to compute the truth value of any formula $p(t_1, ..., t_n)$ in a given structure $\mathcal{S}$ we first have to *evaluate* the terms occurring in it, i.e., to determine the elements of $\mathcal{S}$ denoted by these terms. For that, we must first assign values in $\mathcal{S}$ to the individual variables, by means of a **variable assignment** in $\mathcal{S}$, which is a mapping $v : VAR \to |\mathcal{S}|$ from the set of individual variables VAR to the domain of $\mathcal{S}$.

The evaluation of a term is now done just like the evaluation of an algebraic expression that we know from primary school: starting with the values of the variables and constant symbols we systematically apply the functions in $\mathcal{S}$ which interpret the respective functional symbols occurring in the term. That is, when evaluating a term $f(u_1, ..., u_m)$, we first compute, *recursively*, the values of the arguments $u_1, ..., u_m$ and then apply the interpretation of f in $\mathcal{S}$ to these values.

Formally: due to the unique readability of terms, every variable assignment $v : VAR \to |\mathcal{S}|$ in a structure $\mathcal{S}$ can be uniquely extended to a mapping $v^{\mathcal{S}} : TM(\mathcal{L}) \to |\mathcal{S}|$, called *term evaluation*, such that for every n-tuple of terms $t_1, \ldots, t_n$ and an n-ary functional symbol f:

$$v^{\mathcal{S}}(f(t_1, \ldots, t_n)) = f^{\mathcal{S}}(v^{\mathcal{S}}(t_1), \ldots, v^{\mathcal{S}}(t_n))$$

where $f^{\mathcal{S}}$ is the interpretation of f in $\mathcal{S}$.

Thus, once a variable assignment v in the structure $\mathcal{S}$ is fixed, every term t in $TM(\mathcal{L})$ is evaluated into a unique element $v^{\mathcal{S}}(t)$ of $\mathcal{S}$ (or, just $v(t)$ when $\mathcal{S}$ is fixed), called *the value of the term t for the variable assignment v.*

Note, that *the value of a term only depends on the assignment of values to the variables occurring in that term.* That is, if we denote by $\mathrm{VAR}(t)$ the set of variables occurring in the term t and v_1, v_2 are variable assignments in $\mathcal{S}$ such that $v_1 \mid_{\mathrm{VAR}(t)} = v_2 \mid_{\mathrm{VAR}(t)}$, then $v_1^{\mathcal{S}}(t) = v_2^{\mathcal{S}}(t)$.

Examples:

- Let v be a variable assignment in the structure $\mathcal{N}$ such that $v(x) = 3$ and $v(y) = 5$. Here is a step-by-step computation of the value of the term $s(s(x) \times y)$:

  $v^{\mathcal{N}}(s(s(x) \times y)) = s^{\mathcal{N}}(v^{\mathcal{N}}(s(x) \times y)) = s^{\mathcal{N}}(v^{\mathcal{N}}(s(x)) \times^{\mathcal{N}} v^{\mathcal{N}}(y)) = s^{\mathcal{N}}(s^{\mathcal{N}}(v^{\mathcal{N}}(x)) \times^{\mathcal{N}} v^{\mathcal{N}}(y)) = s^{\mathcal{N}}(s^{\mathcal{N}}(3) \times^{\mathcal{N}} 5) = s^{\mathcal{N}}((3+1) \times^{\mathcal{N}} 5) = ((3+1) \times 5) + 1 = 21.$
- Likewise, $v^{\mathcal{N}}(\mathbf{1} + (x \times s(s(\mathbf{2})))) = 13.$
- If $v(x) =$'Mary' then $v^{\mathcal{H}}(\mathbf{f}(\mathbf{m}(x))) =$ 'the father of the mother of Mary'.

2.2.3 Truth of first-order formulae

Eventually, we want to define the notion of a *formula A to be true in a structure $\mathcal{S}$ for a variable assignment* v, denoted

$$\mathcal{S}, v \models A,$$

compositionally on the structure of the formula A.

Atomic formulae

We begin with the simplest case, where A is an atomic formula. The truth value of the atomic formula $p(t_1, ..., t_n)$ is determined by the interpretation of the predicate symbol p in $\mathcal{S}$, applied to the tuple of arguments $v^{\mathcal{S}}(t_1), \dots, v^{\mathcal{S}}(t_n)$:

$$\mathcal{S}, v \models p(t_1, \dots, t_n) \text{ iff } p^{\mathcal{S}} \text{ holds true for } v^{\mathcal{S}}(t_1), \dots, v^{\mathcal{S}}(t_n).$$

If $\mathcal{S}, v \models p(t_1, \dots, t_n)$ does not hold, we write $\mathcal{S}, v \not\models p(t_1, \dots, t_n)$.

Examples: If L is a binary predicate symbol interpreted in $\mathcal{N}$ as '*less than*', and the variables x and y are assigned values as above, then:

- $\mathcal{N}, v \models L(\mathbf{1} + (x \times s(s(\mathbf{2}))), s(s(x) \times y))$ iff $L^{\mathcal{N}}((\mathbf{1} + (x \times s(s(\mathbf{2}))))^{\mathcal{N}}, (s(s(x) \times y))^{\mathcal{N}})$ iff $13 < 21$, which is *true.*
- likewise, $\mathcal{N}, v \models \mathbf{8} \times (x + s(s(y))) = (s(x) + y) \times (x + s(y))$ iff

  $(\mathbf{8} \times (x + s(s(y))))^{\mathcal{N}} = ((s(x) + y) \times (x + s(y)))^{\mathcal{N}}$ iff $80 = 81$, which is *false.*

Propositional connectives

The truth values propagate over the propositional connectives according to their truth tables, as in propositional logic:

- $\mathcal{S}, v \models \neg A$ iff $\mathcal{S}, v \not\models A$.
- $\mathcal{S}, v \models (A \wedge B)$ iff $\mathcal{S}, v \models A$ and $\mathcal{S}, v \models B$;
- $\mathcal{S}, v \models (A \vee B)$ iff $\mathcal{S}, v \models A$ or $\mathcal{S}, v \models B$;
- $\mathcal{S}, v \models (A \rightarrow B)$ iff $\mathcal{S}, v \not\models A$ or $\mathcal{S}, v \models B$;
- and likewise for $(A \leftrightarrow B)$.

Quantifiers

Finally, the truth of formulae $\forall x A(x)$ and $\exists x A(x)$ is computed according to the meaning of the quantifiers and the truth values of A:

- $\forall x A(x)$ is true if *every object* $\mathbf{a}$ from the domain of $\mathcal{S}$, assigned as a value of x, *satisfies* (i.e. renders true) the formula A.

  Formally, $\mathcal{S}, v \models \forall x A(x)$ if $\mathcal{S}, v[x := a] \models A(x)$ for every $a \in \mathcal{S}$, where the assignment $v[x := a]$ is obtained from v by re-defining $v(x)$ to be a.

- $\exists x A(x)$ is true if *there is an object* **a** from the domain of $\mathcal{S}$ which, assigned as a value of x, satisfies the formula A.

  Formally, $\mathcal{S}, v \models \exists x A(x)$ if there exists an object $a \in \mathcal{S}$, such that $\mathcal{S}, v[x := a] \models A(x)$.

Computing the truth of first-order formulae

Now, we can (at least theoretically) compute the truth of any first-order formula in a given structure for a given variable assignment, step-by-step, following the logical structure of the formula and applying recursively the respective truth condition for the main connective of the currently evaluated subformula.

It is not difficult to show that the truth of a formula in a given structure for a given variable assignment *only depends on the assignment of values to the variables occurring in that formula*[6]. That is, if we denote by $\mathrm{VAR}(A)$ the set of variables occurring in the formula A and v_1, v_2 are variable assignments in $\mathcal{S}$ such that $v_1 \mid_{\mathrm{VAR}(A)} = v_2 \mid_{\mathrm{VAR}(A)}$, then

$$\mathcal{S}, v_1 \models A \text{ iff } \mathcal{S}, v_2 \models A.$$

Still, note that the truth conditions for the quantifiers given above are not really practically applicable when the structure is infinite, because they require taking into account *infinitely many* variable assignments. Thus, evaluating the truth of a first-order formula in an infinite structure is, generally speaking, an infinite procedure. We will address that issue latter. Nevertheless, we can often perform that infinite procedure as finite, by applying uniform (yet, ad hoc) arguments to the infinitely many arising cases.

Example 37 *Consider the structure $\mathcal{N}$ and a variable assignment v such that $v(x) = 0$, $v(y) = 1$, $v(z) = 2$. Then:*

- $\mathcal{N}, v \models \neg(x > y)$.
- *However:* $\mathcal{N}, v \models \exists x(x > y)$. *Indeed,* $\mathcal{N}, v[x := 2] \models x > y$
- *In fact,* $\mathcal{N}, v \models \exists x(x > y)$ *holds for* any *assignment of value to y, and therefore* $\mathcal{N}, v \models \forall y \exists x(x > y)$.
- *On the other hand,* $\mathcal{N}, v \models \exists x(x < y)$, *but* $\mathcal{N}, v \not\models \forall y \exists x(x < y)$. *Why?*
- *What about* $\mathcal{N}, v \models \exists x(x > y \wedge z > x)$*? Well, this is false.*
- *However, for the same variable assignment in the structure of rationals $\mathcal{Q}$, we have that*

  $\mathcal{Q}, v \models \exists x(x > y \wedge z > x)$.

  Does this hold for every variable assignment in $\mathcal{Q}$?

2.2.4 Evaluation games

There is an equivalent, but somewhat more intuitive, and even possibly more entertaining, way to evaluate the truth of first-order formulae in given structures. It is done by playing

[6]Soon we will strengthen this claim.

a special kind of a 2-player game, called *formula evaluation game*[7]. The two players are called *Verifier* and *Falsifier.*[8]

The game is played on a given first-order structure $\mathcal{S}$, variable assignment v in it, and a formula A, the truth of which is to be evaluated in the structure $\mathcal{S}$ for the assignment v. As suggested by the names of the players, the objective of Verifier in the game is to defend and demonstrate the claim that $\mathcal{S}, v \models A$, while the objective of Falsifier is to attack and refute that claim.

The game goes in rounds, and in each round, exactly one of the players – depending on the current 'game configuration' – has to make a move according to rules that we will specify below, until the game ends. The current game configuration $(\mathcal{S}, w, C)$ consists of the structure $\mathcal{S}$, an assignment w in $\mathcal{S}$, and a formula C (the truth of which is to be evaluated in $\mathcal{S}$ for the assignment w). The *initial configuration* is $(\mathcal{S}, v, A)$. We will identify every such game with its initial configuration.

At every round, the player to make a move, as well as the possible moves, are determined by the main connective of the formula in the current configuration $(\mathcal{S}, w, C)$, by rules that closely resemble the truth definitions for the logical connectives. Here are the rules:

- If the formula C is atomic, the game ends.

  If $\mathcal{S}, w \models C$ then Verifier wins, otherwise Falsifier wins.

- If $C = \neg B$ then Verifier and Falsifier *swap their roles* and the game continues with the configuration $(\mathcal{S}, w, B)$. Swapping the roles means that Verifier wins the game $(\mathcal{S}, w, \neg B)$ iff Falsifier wins the game $(\mathcal{S}, w, B)$ and Falsifier wins the game $(\mathcal{S}, w, \neg B)$ iff Verifier wins the game $(\mathcal{S}, w, B)$.

  Intuition: verifying $\neg B$ is equivalent to falsifying B and vice versa.

- If $C = C_1 \wedge C_2$ then Falsifier chooses $i \in \{1, 2\}$ and the game continues with the configuration $(\mathcal{S}, w, C_i)$.

  Intuition: for Verifier to defend the truth of $C_1 \wedge C_2$ he should be able to defend the truth of *any* of the two conjuncts, so it is up to Falsifier to question the truth of either of them.

- If $C = C_1 \vee C_2$ then Verifier chooses $i \in \{1, 2\}$ and the game continues with the configuration $(\mathcal{S}, w, C_i)$.

  Intuition: for Verifier to defend the truth of $C_1 \vee C_2$ it suffices to be able to defend the truth of at least one of the two disjuncts, so he can choose which one.

- If $C = C_1 \rightarrow C_2$ then Verifier chooses $i \in \{1, 2\}$ and, depending on that choice, the game continues respectively with the configuration $(\mathcal{S}, w, \neg C_1)$ or $(\mathcal{S}, w, C_2)$.

  Intuition: $C_1 \rightarrow C_2 \equiv \neg C_1 \vee C_2$.

- If $C = \exists x B$ then Verifier chooses an element $a \in \mathcal{S}$ and the game continues with the configuration $(\mathcal{S}, w[x := a], B)$.

  Intuition: from the truth definition of $\exists x B$: verifying that $\mathcal{S}, w \models \exists x B$ amounts to

[7]Also known as *model checking game.*

[8]Also known by various other names, e.g. Proponent and Opponent, Eloise and Abelard, Eve and Adam. For the sake of convenience, and without prejudice, here we will assume that both players are male.

verifying that $\mathcal{S}, w[x := a] \models B$ for some suitable element $a \in \mathcal{S}$.

- If $C = \forall x B$ then Falsifier chooses an element $a \in \mathcal{S}$ and the game continues with the configuration $(\mathcal{S}, w[x := a], B)$.

  Intuition: from the truth definition of $\forall x B$: falsifying $\mathcal{S}, w \models \forall x B$ amounts to falsifying $\mathcal{S}, w[x := a] \models B$ for some suitable element $a \in \mathcal{S}$.

It is easy to see that any formula evaluation game always ends in a finite number of steps, because the number of logical connectives in the formula in the current configuration strictly decreases after every move, until an atomic formula is reached. It is also obvious that the game always ends with one of the players winning. Now, clearly the winner of such game depends not only on the truth or falsity of the claim $\mathcal{S}, v \models A$, but also on how well the players play the game, so we suppose they always play a best possible move. Thus, the game will be won by the player who *has a winning strategy* for that game, i.e., a rule that, for every possible configuration from which that player is to move assigns such a move, that he is *guaranteed to eventually wins the game, no matter how the other player plays.* It is not quite obvious that one of the players is sure to have a winning strategy in every such game, but it follows from a more general result in game theory. That also follows from the following theorem – which we take for granted – relating the existence of a winning strategy to the truth of the formula in the initial configuration.

Theorem 38 *For every configuration $(\mathcal{S}, v, A)$:*

1. *$\mathcal{S}, v \models A$ iff Verifier has a winning strategy for the evaluation game $(\mathcal{S}, v, A)$.*
2. *$\mathcal{S}, v \not\models A$ iff Falsifier has a winning strategy for the evaluation game $(\mathcal{S}, v, A)$.*

Example 39 *Consider the structure $\mathcal{N}$ and the variable assignment v such that $v(x) = 0$, $v(y) = 1$, $v(z) = 2$.*

1. *Verifier has a winning strategy for the game $(\mathcal{N}, v, \forall y \exists x(x > y + z))$.*

   Indeed, the first move of the game is by Falsifier, who has to choose an integer n and the game continues from configuration $(\mathcal{N}, v[y := n], \exists x(x > y + z))$.

   Now, Verifier has to choose an integer m. For every given $n \in \mathcal{N}$ Verifier can choose e.g., $m = n + 3$. Then he wins the game $(\mathcal{N}, v[y := n][x := m], (x > y + z))$ because $n + 3 > n + 2$.

   Thus, he has a winning strategy for the game $(\mathcal{N}, v[y := n], \exists x(x > y + z))$, for any $n \in \mathcal{N}$.

   Hence, he has a winning strategy for the game $(\mathcal{N}, v, \forall y \exists x(x > y + z))$.

   Therefore, $\mathcal{N}, v \models \forall y \exists x(x > y + z)$.

   In fact, the above winning strategy for Verifier is easy to generalize for any *assignment of value to z, thus showing that $\mathcal{N}, v \models \forall z \forall y \exists x(x > y + z)$.*

2. *Verifier has a winning strategy for the game $(\mathcal{N}, v, \forall x(y < x \vee x < z))$.*

   Indeed, the first move in the game is by Falsifier, who has to choose an integer n and the game continues from configuration $(\mathcal{N}, v[x := n], (y < x \vee x < z))$ in which Verifier must choose one of the disjuncts $y < x$ and $x < z$.

The strategy for Verifier goes as follows: if Falsifier has chosen $n > 1$ *then Verifier chooses the disjunct* $y < x$ *and wins the game* $(\mathcal{N}, v[x := n], y < x)$*; if Falsifier has chosen* $n \leq 1$ *then Verifier chooses the disjunct* $x < z$ *and wins the game* $(\mathcal{N}, v[x := n], x < z)$.

Therefore, $\mathcal{N}, v \models \forall x(y < x \vee x < z)$.

3. *Falsifier has a winning strategy for the game* $(\mathcal{N}, v, \forall x(x < z \to \exists y(y < x)))$.

   Indeed, let Falsifier choose 0 *in the first move. Then the game continues from configuration* $(\mathcal{N}, v[x := 0], (x < z \to \exists y(y < x)))$ *and now Verifier is to choose the antecedent or the consequent of the implication.*

   If Verifier chooses the antecedent, then the game continues from configuration $(\mathcal{N}, v[x := 0], \neg(x < z))$ *which is won by Falsifier because the game* $(\mathcal{N}, v[x := 0], x < z)$ *is won by Verifier, since* $0 < 2$.

   If Verifier chooses the consequent of the implication, then the game continues from configuration $(\mathcal{N}, v[x := 0], \exists y(y < x))$ *and Verifier is to choose a suitable value for* y*. But, whatever* $n \in \mathcal{N}$ *Verifier chooses, he loses the game* $(\mathcal{N}, v[x := 0][y := n], y < x)$ *because* $n < 0$ *is false for every* $n \in \mathcal{N}$.

   Thus, the Verifier has no winning move after the first move of Falsifier choosing 0 *as a value for* x*. Therefore,* $\mathcal{N}, v \not\models \forall x(x < z \to \exists y(y < x))$.

   Furthermore, the winning strategy for Falsifier in the game $(\mathcal{N}, v, \forall x(x < z \to \exists y(y < x)))$ *is a winning strategy for Verifier in the game* $(\mathcal{N}, v, \neg\forall x(x < z \to \exists y(y < x)))$*, hence* $\mathcal{N}, v \models \neg\forall x(x < z \to \exists y(y < x))$.

2.2.5 Translating first-order formulae to natural language

First-order formulae formalize statements about first-order structures and these statements can be translated back to natural language. Of course, a formula is just a string of symbols, so a formal translation could simply consist in writing all symbols in words. That, however, would be of no use for *understanding the meaning* of the formula. While evaluating the truth of the formula in a given structure *formally* does not require translating that formula to natural language and understanding its intuitive meaning, that is essential for practically carrying out the truth evaluation procedure. Indeed, the meaning of (the statement expressed by) a first-order formula in a given interpretation is closely related to its truth in that interpretation. In fact, one can argue that understanding the logical meaning of a formula, spelled out in a natural language, and determining its truth are the two sides of the same coin. In particular, a sensible translation of the formula would surely help the player who has a winning strategy in the evaluation game to come up with the right strategy, and thus to establish the truth of the formula in the given interpretation.

So, let us look at some examples of translating to natural language and evaluating the truth of first-order formulae, now interpreted in $\mathcal{H}$. Note that a good translation is usually not word-by-word, but it needs some polishing and rephrasing in the target language so it eventually sounds naturally and makes good sense.

1.
$$\mathsf{John} = \mathsf{f}(\mathsf{Mary}) \wedge \mathsf{L}(\mathsf{John}, \mathsf{Mary})$$
— “John is the father of Mary and he loves her”.

2.
$$(\mathsf{John} = \mathsf{f}(\mathsf{Mary})) \to \exists x \mathsf{L}(x,\ \mathsf{Mary})$$
— “If John is the father of Mary then (there is) someone (who) loves Mary”.

3.
$$\exists x \forall z(\neg \mathsf{L}(z, y) \to \mathsf{L}(x, z))$$
— “There is someone (x) who loves everyone who does not love y”.

(Note that y stands for an unspecified person here, so this is not a proposition.)

4.
$$\forall x(\exists y(x = \mathsf{m}(y)) \to \forall z(x = \mathsf{m}(z) \to \mathsf{L}(x, z))).$$
— “Every mother loves all her children.”

2.2.6 Exercises

1. Evaluate the occurring terms and determine the truth of the following atomic formulae in $\mathcal{R}$:
   a) $\times(s(s(x)), \mathbf{2}) = \times(x, s(s(\mathbf{2})))$, where x is assigned value 3.
   b) $+(\mathbf{5}, \times(x, +(\mathbf{3}, y))) > s(\times(\mathbf{3}, \times(\mathbf{2}, x)))$, where x is assigned value $\frac{1}{2}$ and y is assigned value -5.
   c) $(s(\times(\times(x, x), x))) < \times(s(x), +(\times(x, x), -(\ \mathbf{1}, x)))$, for each of the following values of x: 2, $\sqrt{2}$, π.
2. Determine the first-order instances of the propositional formula $A = (\neg p \to (q \vee \neg(p \wedge q)))$ where:
   a) $\neg(f(x) = y)$ is substituted for p and $\exists x(x > y)$ is substituted for q.
   b) $\neg(x = y)$ is substituted both for p and q.
3. Is the formula $(\neg(x = y) \to (\exists x(x > y) \vee \neg((y = x) \wedge \exists x(x > y))))$ a first-order instance of the propositional formula A from the previous exercise?
4. Show, by induction on the length of terms in any given first-order language $\mathcal{L}$, that if v_1, v_2 are variable assignments in $\mathcal{S}$ such that $v_1 \mid_{\mathrm{VAR}(t)} = v_2 \mid_{\mathrm{VAR}(t)}$, then $v_1^{\mathcal{S}}(t) = v_2^{\mathcal{S}}(t)$.
5. Show, by induction on the length of formulae in any given first-order language $\mathcal{L}$, that, if v_1, v_2 are variable assignments in $\mathcal{S}$ such that $v_1 \mid_{\mathrm{VAR}(A)} = v_2 \mid_{\mathrm{VAR}(A)}$, then $\mathcal{S}, v_1 \models A$ iff $\mathcal{S}, v_2 \models A$.
6. Consider the structure $\mathcal{N}$ and a variable assignment v such that $v(x) = 1$, $v(y) = 2$, $v(z) = 3$. Determine the truth in $\mathcal{N}$ under v of each of the following formulae:
   a) $z > y \to x > y$.

b) $\exists x(z > y \to x > y)$.

c) $\exists y(z > y \to x > y)$.

d) $\exists z(z > y \to x > y)$.

e) $\forall y(z > y \to x > y)$.

f) $\exists x \forall y(z > y \to x > y)$.

g) $\exists x \forall z(z > y \to x > y)$.

h) $\exists z \forall y(z > y \to x > y)$.

i) $\exists z \forall x(z > y \to x > y)$.

j) $\exists y \forall x(z > y \to x > y)$.

k) $\exists y \forall z(z > y \to x > y)$.

l) $\forall x \exists y(z > y \to x > y)$.

m) $\forall x \exists z(z > y \to x > y)$.

n) $\forall y \exists x(z > y \to x > y)$.

o) $\forall y \exists z(z > y \to x > y)$.

p) $\forall z \exists x(z > y \to x > y)$.

q) $\forall z \exists y(z > y \to x > y)$.

r) $\exists x \forall y \forall z(z > y \to x > y)$.

s) $\exists y \forall x \forall z(z > y \to x > y)$.

t) $\exists z \forall x \forall y(z > y \to x > y)$.

u) $\forall y \exists x \forall z(z > y \to x > y)$.

v) $\forall x \exists y \forall z(z > y \to x > y)$.

w) $\forall z \exists x \forall y(z > y \to x > y)$.

x) $\forall x \forall y \exists z(z > y \to x > y)$.

y) $\forall y \forall z \exists x(z > y \to x > y)$.

z) $\forall z \forall x \exists y(z > y \to x > y)$.

7. Translate into English (or Danish) the following first-order formulae and determine which of them represent true propositions when interpreted in $\mathcal{R}$.

a) $\neg \forall x(x \neq 0)$;

b) $\forall x(x^3 \geq x)$;

c) $\forall x(x = x^2 \to x > \mathbf{0})$;

d) $\exists x(x = x^2 \wedge x < \mathbf{0})$;

e) $\exists x(x = x^2 \to x < \mathbf{0})$;

f) $\forall x(x > \mathbf{0} \to x^2 > x)$;

g) $\forall x(x = \mathbf{0} \vee \neg x + x = x)$;

h) $\forall x((x = x^2 \land x > 1) \to x^2 < 1)$;

i) $\forall x \forall y(x > y \lor y > x)$;

j) $\forall x \exists y(x > y^2)$;

k) $\forall x \exists y(x > y^2 \lor y > \mathbf{0})$;

l) $\forall x(x \geq \mathbf{0} \to \exists y(y > \mathbf{0} \land x = y^2))$;

m) $\forall x \exists y(x > y \to x > y^2)$;

n) $\forall x \exists y(\neg x = y \to x > y^2)$;

o) $\exists x \forall y(x > y)$;

p) $\exists x \forall y(x + y = x)$;

q) $\exists x \forall y(x + y = y)$;

r) $\exists x \forall y(x > y \lor -x > y)$;

s) $\exists x \forall y(x > y \lor \neg x > y)$;

t) $\exists x \forall y(y > x \to y^2 > x)$;

u) $\exists x \forall y(x > y \to x > y^2)$;

v) $\exists x \exists y(xy = x + y)$;

w) $\forall x \exists y \forall z(xy = yz)$;

x) $\exists x \forall y \exists z((x + y)z = 1)$;

y) $\forall x \forall y(x > y \to \exists z(x > z \land z > y))$;

z) $\forall x \exists z \forall y(x > z \to z > y)$.

2.3 Basic grammar and use of first-order languages

The best (and only) way to learn a new language is by using it; in particular, by practicing translation between that language and one that we know well. In this section we will discuss how to use first-order languages. For that, we will have to understand their basic grammar, but first we will look at some examples on translating statements from natural language to first-order languages.

2.3.1 Translation from natural language to first-order languages: warming up

1. Let us start with translating

   "There is an integer greater than 2 and less than 3."

   Surely, this is false, but now we are not concerned with its truth, only with its translation to $\mathcal{L}_{\mathcal{Z}}$. It is immediate:

   $$\exists x(x > \mathbf{2} \wedge x < \mathbf{3}).$$

   Note that **2** and **3** here are *terms*, not the numbers 2 and 3.

2. Does the same translation work for $\mathcal{L}_{\mathcal{Q}}$? No, because the quantification is over *all elements of the domain of discourse*, so the same formula would say in $\mathcal{L}_{\mathcal{Q}}$ "There is a rational number greater than 2 and less than 3", which is true, but not what we wanted to say. So, what now? Well, if we want to refer to *integers* while the domain of discourse is a larger set, viz. the rational numbers, we need to extend the language with an additional *predicate I,* which says that the element of the domain *is an integer.* Now, the translation is easy:

   $$\exists x(I(x) \wedge x > \mathbf{2} \wedge x < \mathbf{3}).$$

   We will come back to this trick later again.

3. Let us now translate

   "There is no real number the square of which equals -1 ."

   to $\mathcal{L}_{\mathcal{R}}$. How about

   $$\exists x(\neg x^2 = -1)?$$

   Although the sentence begins with "There is...", do not be confused: this is *not an existential statement, but a negation of one* . It actually says

   "It is not true that there is a real number the square of which equals -1."

   so the correct translation is

   $$\neg\exists x(x^2 = -1).$$

2.3.2 Restricted quantification

Now, take again the sentence

"Every man loves a woman."

In order to formalize it in the language of first-order logic, we need some preparation. It looks deceptively simple, yet it presents a problem we have already seen before: the quantifiers in our formal language range over the *whole* domain of discourse, in our case over all human beings. So, whenever we write $\forall x$, in this context it will mean 'every human being', not 'every man'; likewise, using $\exists x$ we say 'there is a human being', but we cannot say 'there is a woman'. On the other hand, we usually quantify not over all individuals from the domain, but over a specified family of them, e.g. "there is a *positive* integer x such that ...", "Every *child* ...", etc. To resolve this problem we use (again) a little trick with the predicates, called **restricted quantification**. Thinking a little on the sentence we realize that it can be rephrased in the following, somewhat awkward but more convenient to formalize in first-order logic, way: "For every human, if he is a man then there is a human who is a woman and the man loves that woman". Now the translation into $\mathcal{L}_{\mathcal{H}}$ is immediate:

$$\forall x(\mathsf{M}(x) \rightarrow \exists y(\mathsf{W}(y) \wedge \mathsf{L}(x, y))).$$

In general, we introduce unary predicates for the type of objects we want to quantify over, and use logical connectives to express the *restricted quantification schemes*, as we did in the example above. For instance, in order to quantify over positive numbers in the domain of all integers, we can introduce a predicate $P(x)$, saying that "x is a positive number" and then write:

$$\exists x(P(x) \wedge \ldots x \ldots)$$

saying that there exists an object x *which is a positive number* and which satisfies $\ldots x \ldots$. Likewise,

$$\forall x(P(x) \rightarrow \ldots x \ldots)$$

says that all objects x , *which are positive numbers*, satisfy $\ldots x \ldots$.

Note, that sometimes we do not really need to introduce a new predicate in the language, if it is already *definable* there. For instance, in the latter case, the predicate for (or, the set of) positive numbers is definable in the language $\mathcal{L}_{\mathcal{Z}}$ on the structure $\mathcal{Z}$ as $\mathbf{0} < x$ Thus, the schemes $\exists x(\mathbf{0} < x \wedge \ldots x \ldots)$ and $\forall x(\mathbf{0} < x \rightarrow \ldots x \ldots)$ will have the same effect as those above.

An alternative way to formalize restricted quantification is to use, as suggested earlier, *many-sorted domains*. For instance, in the domain of real numbers we can have sorts for integers, rational numbers, etc.; in the domain of humans — sorts for male, female, child, etc.. Respectively, we would need different, and disjoint, stocks of variables for each sort[9]. While the former method is more commonly used in informal reasoning, the latter is a more universal one. In the usual mathematical practice, though, the following convenient

[9]In fact, this is usually done in mathematical discourse and writings. For instance, it is a tradition to denote a real number by x, y, z etc., while an integer would rather be denoted by i, j, k, n, m, a function by f, g, h, etc. Of course, there is no inherent reason, apart from the risk of confusion, not to use f for an integer, or j for a real number. And, when working with complex numbers, i becomes something quite different...

combination of the two methods is used. For the most important families of individuals, we introduce some standard notation, e.g. the set of natural numbers is usually denoted by $\mathbf{N}$, the set of integers by $\mathbf{Z}$, the set of rational numbers by $\mathbf{Q}$, and the set of reals by $\mathbf{R}$. Now, if we want to quantify e.g. over integers, we can say: "*for all x in* $\boldsymbol{Z}$ ", respectively " *there is x in* $\boldsymbol{Z}$ *such that* ", or symbolically: $\forall x \in \mathbf{Z}(\ldots x \ldots)$, and $\exists x \in \mathbf{Z}(\ldots x \ldots)$. For instance, $\forall x \in \mathbf{Z}(x^2 \geq x)$ states (truly) that the square of every *integer* is greater than or equal to that integer; $\exists x \in \mathbf{Q}(x^3 - 2x + 2 = \mathbf{0})$ states (falsely) that there is a *rational* solution of the equation $x^3 - 2x + 2 = \mathbf{0}$, and $\forall x \in \mathbf{R} \exists z \in \mathbf{Z}(x < z)$ says that for every real number there is an integer greater than it. (True or false?).

2.3.3 Free and bound variables. Scope of a quantifier.

There are two essentially different ways in which we use individual variables in first-order formulae.

1. First, we use them to denote *unknown or unspecified objects*, like in

$$(\mathbf{5} < x) \vee (x^2 + x - \mathbf{2} = \mathbf{0}).$$

   We say that the variable x occurs **free** in that formula, or simply that x is a **free variable** in it. As long as a formula contains free variables, it cannot (in general) be assigned a truth-value until these free variables are assigned values. Therefore, a formula containing free variables cannot be regarded as a proposition, for its truth may vary depending on the different possible values of the occurring free variables. For example, assigning value 1 or 6 to x in the formula above turns it into a true proposition in $\mathcal{R}$, while assigning value 2 to x turns it into a false one.

2. Second, we use individual variables in order *to quantify* over individuals, as in

$$\exists x((\mathbf{5} < x) \vee (x^2 + x - \mathbf{2} = \mathbf{0})) \text{ and } \forall x((\mathbf{5} < x) \vee (x^2 + x - \mathbf{2} = \mathbf{0})).$$

   In these formulae the variable x is said to occur **bound** (or simply, to be a bound variable): by the quantifier $\forall$ (*universally bound*) in the formula on the left, and respectively by $\exists$ (*existentially bound*) in the one on the right.

Note that the same variable can be *both free and bound* in a formula. For example: x in the formula $\mathbf{0} < x \wedge \exists x(\mathbf{5} < x)$. That is why we talk about free and bound *occurrences* of a variable in a formula.

To make the notion of a bound occurrence of a variable more precise, note that every occurrence of a quantifier Q in a formula is the beginning of a *unique* subformula QxA, called **the scope** of that occurrence of the quantifier. For example, in

$$\forall x((x > \mathbf{5}) \rightarrow \forall y(y < \mathbf{5} \rightarrow (y < x \wedge \exists x(x < \mathbf{3}))))$$

the scope of the first occurrence of $\forall$ is the whole formula, while the scope of the second occurrence of $\forall$ is $\forall y(y < \mathbf{5} \rightarrow (y < x \wedge \exists x(x < \mathbf{3})))$, and the scope of the occurrence of $\exists$ is $\exists x(x < \mathbf{3})$.

Now, every bound occurrence of a variable x is bound by the *innermost* occurrence of a quantifier Q over x in which scope that occurrence of x is. In other words, an occurrence of a variable x is bound by the first occurrence of the quantifier Q in a formula QxA, if and only if that occurrence of x is free in the subformula A. For example, in the formula above, the first three occurrences of x are bound by the first occurrence of $\forall$, while the last two are bound by the occurrence of $\exists$.

Definition 40 *A formula with no bound variables is called an* ***open formula****. A formula with no free variables is called a* ***closed formula****, or a* ***sentence****.*

Once interpreted in a structure, sentences have determined meaning and represent propositions about that structure, the truth of which does not depend on a variable assignment. More generally, the following holds.

Proposition 41 *The truth of any formula in a given structure for given variable assignment only depends on the assignment of values to the variables occurring* free *in that formula. That is, if we denote by* $\mathrm{FV}(A)$ *the set of free variables in the formula A, and v_1, v_2 are variable assignments in $\mathcal{S}$ such that* $v_1 \mid_{\mathrm{FV}(A)} = v_2 \mid_{\mathrm{FV}(A)}$, *then*

$$\mathcal{S}, v_1 \models A \textit{ iff } \mathcal{S}, v_2 \models A.$$

2.3.4 Renaming of a bound variable in a formula. Clean formulae.

Note that a bound variable in a formula plays an auxiliary rôle and *does not have its own meaning* (we also call it a *dummy* variable) in the sense that we can replace it by another one throughout the formula without altering the meaning of that formula. For example, it should be intuitively clear that $\exists x(5 < x \vee x^2 + x - \mathbf{2} = 0)$ means exactly the same as $\exists y(\mathbf{5} < y \vee y^2 + y - \mathbf{2} = \mathbf{0})$ (in particular, these two are equally true); likewise $\forall x(\mathbf{5} < x \vee x^2 + x - \mathbf{2} = \mathbf{0})$ means the same as $\forall y(\mathbf{5} < y \vee y^2 + y - \mathbf{2} = \mathbf{0})$. On the other hand, $\mathbf{5} < x \vee x^2 + x - \mathbf{2} = \mathbf{0}$ is *essentially different* from $\mathbf{5} < y \vee y^2 + y - \mathbf{2} = \mathbf{0}$ – depending on the values of the free variables x and y, one of these can be true, while the other one is false.

You should distinguish very well between the meaning and the use of free and bound variables. For example, the free and the bound occurrences of the variable x in the formula $(x > \mathbf{5}) \wedge \forall x(x < \mathbf{2}x)$ have *nothing* to do with each other.

Furthermore, different occurrences of the same variable can be bound by different quantifiers, and that may be confusing, too. Examples:

- $\exists x(x > \mathbf{5}) \vee \forall x(\mathbf{2}x > x)$. Clearly, the occurrences of x, bound by the first quantifier, have nothing to do with those bound by the second one.
- $\exists x(x > \mathbf{5}) \wedge \exists x(x < \mathbf{3})$. Likewise, the two x's claimed to exist here need not (and, in fact, cannot) be the same.
- $\forall x((x > \mathbf{5}) \rightarrow \exists x(x < \mathbf{3}))$. Again, the occurrences of x in the subformula $\exists x(x < \mathbf{3})$ are bound by $\exists$ and not related to the first two occurrences of x, bound by $\forall$.

The best way to avoid confusions like these in your formal mathematical arguments and proofs is *always to use different variables for different purposes.* In particular, never use the same variable as both free and bound, nor as bound by two different quantifiers in the same formula or proof.

Definition 42 *A formula A is* ***clean*** *if no variable occurs both free and bound in A and every two occurrences of quantifiers bind different variables.*

Thus, $\exists x(x > \mathbf{5}) \wedge \exists y(y < z)$ is clean, while $\exists x(x > \mathbf{5}) \wedge \exists y(y < x)$ and $\exists x(x > \mathbf{5}) \wedge \exists x(y < x)$ are not.

Definition 43 *The uniform replacement of all occurrences of a variable x bound by the same occurrence of a quantifier in a formula A with a variable not occurring in A is called a* ***renaming*** *of the variable x in A.*

It should be evident that every formula can be transformed into a clean formula by means of several consecutive renamings of variables. For example, the formula

$$(x > \mathbf{5}) \wedge \forall x((x > \mathbf{5}) \to \neg\exists x(x < y))$$

can be transformed into the clean formula

$$(x > \mathbf{5}) \wedge \forall z_1((z_1 > \mathbf{5}) \to \neg\exists z_2(z_2 < y)).$$

2.3.5 Substitution of a term for a variable in a formula. Capture of a variable.

Given a formula A, a variable x, and a term t, we can obtain a formula $A[t/x]$ by substituting *simultaneously* t for all *free* occurrences of x in A. The formula $A[t/x]$ is called **the result of substitution of t for x in A**. The semantic meaning of such substitution is to assign the value of the term t to the variable x when evaluating the truth of A. For instance, given the formula

$$A = \forall x(P(x,y) \to (\neg Q(y) \vee \exists y P(x,y))),$$

we have

$$A[f(y,z)/y] = \forall x(P(x,f(y,z)) \to (\neg Q(f(y,z)) \vee \exists y P(x,y))),$$

while

$$A[f(y,z)/x] = A,$$

because x does not occur free in A.

Intuitively, $A[t/x]$ is supposed to say about the individual denoted by t the same as what A says about the individual denoted by x. But, is that always the case? No. An unwanted *'capture'* effect can occur after such substitution if we substitute a term t for a variable x in a formula A, where x is in the scope of a quantifier over another variable y which occurs in t. For example: substituting $y + \mathbf{1}$ for x in the formula $\exists y(x < y)$, which is true in $\mathcal{N}$ for *every* value assigned to x, will produce the false sentence $\exists y(y+\mathbf{1} < y)$ since the occurrence of y in the term $y + \mathbf{1}$ is *captured* by the quantifier $\exists y$ because the substitution mixed free and the bound uses of y. Similarly, in the formula $\forall x \exists z(x < z))$ meaning that for every number there is a greater one, renaming x with the variable z will produce $\forall z \exists z(z < z)$, which clearly distorts the meaning of the formula. The reason is that when substituting z for x it is *captured* by the quantifier $\exists z$.

Formally, **capture** happens when new occurrences of a variable, say y, are introduced in the scope of a quantifier Qy in a formula A as a result of substitution of a term t containing y for another variable x in that formula. The following definition means to eliminate substitutions that allow capture.

Definition 44 *A term t is* ***free for (substitution for) a variable*** *x in a formula A, if no variable in t is captured by a quantifier as t is substituted for any* free occurrence *of x in A.*

Example 45

- *Every ground term (not containing variables), in particular every constant symbol, is free for substitution for any variable in any formula.*
- *More generally, every term that does not contain variables with bound occurrences in a given formula is free for substitution for any variable in that formula.*
- *The term $f(x, y)$ is free for substitution for y in the formula $A = \forall x(P(x, z) \wedge \exists y Q(y)) \to P(y, z)$, resulting in $A[f(x, y)/y] = \forall x(P(x, z) \wedge \exists y Q(y)) \to P(f(x, y), z)$.*
- *However, the same term is not free for substitution for z in A, resulting in $A[f(x, y)/z] = \forall x(P(x, f(x, y)) \wedge \exists y Q(y)) \to P(y, f(x, y))$, because a capture occurs, of the variable x in the first occurrence of $f(x, y)$.*

To avoid the problems arising from capture, mentioned above, from now on we only allow a substitution t/x in a formula A when t is free for x in A.

2.3.6 A note on renamings and substitutions in a formula

Note that renaming and substitution are *very different* operations: renaming always acts on bound variables, while substitution always acts on free variables.

Also, as we will see later, renamings preserve the formula up to logical equivalence, while substitutions do not.

On the other hand, a suitable renaming of a formula can prepare it for a substitution, by rendering the term to be substituted free for such substitution in the renamed formula. For instance, the term $f(x, y)$ is not free for substitution for y in

$$A = \forall x(P(x, y) \wedge \exists y Q(y)),$$

but it becomes free for such substitution after renaming of A, e.g., to

$$A' = \forall x'(P(x', y) \wedge \exists y Q(y)).$$

2.3.7 Exercises

1. Using the additional predicate $\mathsf{I}(x)$ for 'x is an integer' formalize the following sentences in the first-order language for the structure of real numbers $\mathcal{R}$ and determine the truth value of each of them in $\mathcal{R}$.

   a) Every square of an integer is greater than 0.

   b) Every square of a real number which is not integer is greater than 0.

c) Some real numbers are not integers.
d) Every integer is even or odd.
e) No integer is both even and odd.
f) For every integer there is a greater integer.
g) Every positive integer is a square of some negative real number.
h) Not every real number is greater than an integer.
i) There is an integer such that not every real number is greater than it.
j) No real number is greater than every integer.
k) Every real number which is not zero has a reciprocal.
l) Some real number when multiplied by any real number, produces that number.
m) Between every two different real numbers there is an integer.

2. Using unary predicates $\mathsf{T}(x)$ for 'x talks' and $\mathsf{L}(x)$ for 'x listens', formalize the following sentences in a first-order language for the domain of all humans.

a) Everybody talks or everybody listens.
b) Everybody talks or listens.
c) If John talks everybody listens.
d) If somebody talks everybody listens.
e) If somebody talks everybody else listens.
f) Nobody listens if everybody talks.

3. Translate the following sentences to the first-order language $\mathcal{L}_{\mathcal{H}}$ for the structure of all humans $\mathcal{H}$. Remember that the language $\mathcal{L}_{\mathcal{H}}$ does not have a unary predicate for '...is a child', but a binary predicate '...is a child of...'.

a) Some men love every woman.
b) Every woman loves every man who loves her.
c) Every man loves a woman who does not love him.
d) Some women love no men who love them.
e) Some men love only women who do not love them.
f) No woman loves a man who loves every woman.
g) Every woman loves her children.
h) Every child loves his/her mother.
i) Everyone loves a child. *(Note the ambiguity here...)*
j) A mother loves every child. *(... and here.)*
k) Every child is loved by someone.
l) Some woman loves every child.

4. Determine the scope of each quantifier and the free and bound occurrences of variables in the following formulae (where P is a unary predicate, and Q a binary one.)

   a) $\exists x \forall z (Q(z,y) \lor \neg \forall y (Q(y,z) \to P(x)))$,

   b) $\exists x \forall z (Q(z,y) \lor \neg \forall x (Q(z,z) \to P(x)))$,

   c) $\exists x (\forall z Q(z,y) \lor \neg \forall z (Q(y,z) \to P(x)))$,

   d) $\exists x (\forall z Q(z,y) \lor \neg \forall y Q(y,z)) \to P(x)$,

   e) $\exists x \forall z (Q(z,y) \lor \neg \forall y Q(x,z)) \to P(x)$,

   f) $\exists x \forall z Q(z,y) \lor \neg (\forall y Q(y,x) \to P(x))$,

   g) $\exists x (\forall z Q(z,y) \lor \neg (\forall z Q(y,z) \to P(x)))$,

   h) $\exists x (\forall x Q(x,y) \lor \neg \forall z (Q(y,z) \to P(x)))$,

   i) $\exists x (\forall y (Q(x,y) \lor \neg \forall x Q(x,z))) \to P(x)$.

5. Rename the bound variables in each of the formulae above to obtain a clean formula.

6. For each of the following formulae (where P is a unary predicate, and Q a binary one) determine if the indicated term is free for substitution for the indicated variable. If so, do the substitution.

   a) Formula: $\exists x (\forall z P(y) \lor \neg \forall y (Q(y,z) \to P(x)))$; term: $f(x)$; variable: z.

   b) Same formula; term: $f(z)$; variable: y.

   c) Same formula; term: $f(y)$; variable: y.

   d) Formula: $\forall x ((\neg \forall y Q(x,y) \lor P(z)) \to \forall y \neg \exists z \exists x Q(z,y))$; term: $f(y)$; variable: z.

   e) Same formula; term: $g(x, f(z))$; variable: z.

   f) Formula: $\forall y (\neg (\forall x \exists z (\neg P(z) \land \exists y Q(z,x))) \land (\neg \forall x Q(x,y) \lor P(z)))$; term $f(x)$; variable: z.

   g) Same formula; term: $f(y)$; variable: z.

   h) Formula: $(\forall y \exists z \neg P(z) \land \forall x Q(z,x)) \to (\neg \exists y Q(x,y) \lor P(z))$; term: $g(f(z),y)$; variable: z.

   i) Same formula; term: $g(f(z),y)$; variable: x.

7. For each of the following formulae identify the scopes of all occurrences of quantifiers and identify the free and the bound occurrences of variables in A.

   Then, for each of x, y and z determine, with reason, whether the term $f(x,z)$ is free for substitution for that variable in A; whenever this is the case, do the substitution.

   a) $\forall x (\neg \forall y Q(x,y) \land P(z)) \to \exists z (\forall y Q(z,y) \land \neg P(x))$

   b) $\neg \exists x (\forall y (\exists z Q(y,z) \leftrightarrow P(z)) \land \forall z R(x,y,z))$

   c) $\neg (\forall y (\forall z Q(y,z) \to P(z)) \to \exists z (P(z) \land \forall x (Q(z,y) \to Q(x,z))))$

   d) $\neg (\neg \exists z (P(z) \land \forall x (Q(z,y) \to Q(x,z))) \to \neg \forall y (\forall z Q(y,z) \to P(z)))$

   e) $\neg (\forall y (\neg \exists z Q(y,z) \to P(z)) \to \exists z ((P(z) \to Q(z,y)) \land \neg \exists x R(x,y,z)))$

2.4 Logical validity, equivalence, and consequence

Here we will introduce and discuss the fundamental logical notions of logical validity, equivalence, and consequence for first-order formulae.

2.4.1 Truth of first-order sentences in structures. Models and counter-models.

Recall that a sentence is a formula with no free variables.

As we noted earlier, the truth of a sentence in a given structure *does not depend on the variable assignment.* Therefore, for a structure $\mathcal{S}$ and sentence A we can simply write

$$\mathcal{S} \models A$$

if $\mathcal{S}, v \models A$ for *any* (hence, *every*) variable assignment v.

We then say that $\mathcal{S}$ **is a model of** A and that A **is true in** $\mathcal{S}$, or that A **is satisfied by** $\mathcal{S}$. Otherwise we write $\mathcal{S} \not\models A$ and say that A **is false in** $\mathcal{S}$, or that $\mathcal{S}$ **is a counter-model for** A.

Example 46 *True sentences express properties of structures, for example:*

- $\mathcal{R} \models \forall x \forall y (x + y = y + x)$ *(commutativity of the addition of real numbers);*

  $\mathcal{R} \models \forall x \forall y \forall z ((x+y) \times z = x \times z + y \times z)$ *(distributivity of multiplication over addition of real numbers)*
- $\mathcal{N} \models \forall x \exists y (x < y)$ *(for every natural number there is a larger one),*

  while $\mathcal{N} \not\models \forall x \exists y (y < x)$ *(not for every natural number there is a smaller one);*

  hence, $\mathcal{N} \models \neg \forall x \exists y (y < x)$.

  However, $\mathcal{Z} \models \forall x \exists y (y < x)$.
- $\mathcal{Q} \models \forall x \forall y (x < y \to \exists z (x < z \land z < y))$ *(the ordering of the rationals is dense),*

  while $\mathcal{Z} \not\models \forall x \forall y (x < y \to \exists z (x < z \land z < y))$ *(the ordering of the integers is not dense).*
- $\mathcal{Q} \models \exists x (3x = 1)$, *but* $\mathcal{Z} \not\models \exists x (3x = 1)$ *(*$1/3$ *is a rational number, but is not an integer).*
- $\mathcal{R} \models \forall x (x > \mathbf{0} \to \exists y (y^2 = x))$ *(every positive real number is a square of a real number),*

  but $\mathcal{Q} \not\models \forall x (x > \mathbf{0} \to \exists y (y^2 = x))$ *(not every positive rational number is a square of a rational number),*
- $\mathcal{R} \not\models \forall x \forall y (xy > \mathbf{0} \to (x > \mathbf{0} \lor y > \mathbf{0}))$. *(Take e.g.,* $x = y = -1$*).*
- $\mathcal{R} \models \exists x \forall y (xy < \mathbf{0} \to y = \mathbf{0})$*?*
- $\mathcal{H} \models \forall x \forall y (\exists z (x = \mathsf{m}(z) \land y = \mathsf{m}(z)) \to x = y)$.

  (Is the above sentence only true for this interpretation?)

- $\mathcal{N} \models (\forall x(\forall y(y < x \rightarrow P(y)) \rightarrow P(x)) \rightarrow \forall x P(x))$,

  where P is any (uninterpreted) unary predicate.

  Why is this true? What does this sentence say about $\mathcal{N}$? Is it true in $\mathcal{Z}$, too?

2.4.2 Satisfiability and validity of first-order formulae

A **first-order formula** A **is:**

- **satisfiable** if $\mathcal{S}, v \models A$ for *some* structure $\mathcal{S}$ and *some* variable assignment v in $\mathcal{S}$.
- **(logically) valid**, denoted $\models A$, if $\mathcal{S}, v \models A$ for *every* structure $\mathcal{S}$ and *every* variable assignment v in $\mathcal{S}$.

If A is a sentence, the variable assignment in these definitions is not relevant and can be omitted. Actually, validity of any formula can be reduced to a validity of a sentence, as follows.

Definition 47 *Given a first-order formula A with free variables $x_1, ..., x_n$, the sentence $\forall x_1 ... \forall x_n A$ is called a* ***universal closure*** *of A.*

Note that, depending on the order of the quantified variables, a formula can have several universal closures but, as we will see further, they are all essentially equivalent. The following is an easy exercise to prove.

Proposition 48 *A formula A is logically valid if and only if any of its universal closures is logically valid.*

Example 49 *Some examples of logically valid and non-valid first-order formulae:*

- *Every first-order instance of a tautology is logically valid.*

  Thus, for instance, $\models \neg\neg(x > \mathbf{0}) \rightarrow (x > \mathbf{0})$ and $\models P(x) \vee \neg P(x)$, for any unary predicate P.

- *If A is a logically valid formula and x is any variable then $\forall x A$ is valid, too.*

  Thus, $\models \forall x(P(x) \vee \neg P(x))$, as well as $\models \forall x(P(y) \vee \neg P(y))$.

- $\models \forall x(x = x)$, *because of the very meaning of the equality symbol* $=$.

- $\models \forall x \forall y(\exists z(x = f(z) \wedge y = f(z)) \rightarrow x = y)$, *where f is any unary functional symbol,*

  because of the very meaning of the notion of a function.

- *The sentence $\exists x P(x)$ is* not valid*: take for instance $P(x)$ to be interpreted as the empty set in any non-empty domain. This will give a* counter-model *for that sentence. However, $\exists x P(x)$ is* satisfiable*; a model is any structure where the interpretation of $P(x)$ is non-empty.*

- *The sentence $\forall x P(x) \vee \forall x \neg P(x)$ is not valid, but it is* satisfiable.

  (Find a model and a counter-model.)

- *The sentence $\exists x(P(x) \wedge \neg P(x))$ is* not satisfiable. *Why?*

- $\models \exists x \forall y P(x,y) \to \forall y \exists x P(x,y)$.

  It is a good exercise to check this from the definition; we will discuss it again later.
- *However,* $\not\models \forall y \exists x P(x,y) \to \exists x \forall y P(x,y)$. *Find a counter-model!*

2.4.3 Logical consequence in first-order logic

The notion of logical consequence is fundamental in logic. In propositional logic we defined logical consequence $A_1, \dots, A_n \models B$ as '*preservation of the truth from the premises to the conclusion*'. The same idea applies in first-order logic, but instead of truth tables, we now use *truth in a structure.*

Definition 50 *A first-order formula B is a* ***logical consequence*** *from the formulae $A_1, \dots, A_n$, denoted $A_1, \dots, A_n \models B$, if for every structure $\mathcal{S}$ and variable assignment v in $\mathcal{S}$ for which the formulae $A_1, \dots, A_n$ are satisfied (equivalently, $A_1 \wedge \dots \wedge A_n$ is satisfied) in $\mathcal{S}$ by v, the formula B is satisfied in $\mathcal{S}$ by v, too. Formally:*

$$\mathcal{S}, v \models A_1 \wedge \dots \wedge A_n \quad \textit{implies} \quad \mathcal{S}, v \models A.$$

In particular, if $A_1, \dots, A_n, B$ are sentences, $A_1, \dots, A_n \models B$ means that B is true in every structure in which all $A_1, \dots, A_n$ are true, that is: every model of each of $A_1, \dots, A_n$ is a model of of B, too.

If $A_1, \dots, A_n \models B$, we also say that B ***follows logically from*** *$A_1, \dots, A_n$, or that $A_1, \dots, A_n$* ***logically imply*** *B.*

Note that:

- $\emptyset \models A$ iff $\models A$.
- $A_1, \dots, A_n \models B$ if and only if $\models A_1 \wedge \dots \wedge A_n \to B$.

Thus, logical consequence in first-order logic is reduced to logical validity.

Example 51 *Some examples and non-examples of first-order logical consequences:*

1. *If $A_1, \dots, A_n, B$ are propositional formulae such that $A_1, \dots, A_n \models B$, and $A'_1, \dots, A'_n, B'$ are first-order instances of $A_1, \dots, A_n, B$ obtained by the same substitution, then $A'_1, \dots, A'_n \models B'$.*

   For instance, $\exists x A$, $\exists x A \to \forall y B \models \forall y B$.
2. $\forall x A(x) \models A[t/x]$ *for any formula A and term t free for x in A.*

   Indeed, if $\forall x A(x)$ is true in a structure $\mathcal{S}$, for some assignment, then A is true for every possible value of x in $\mathcal{S}$, including the value of t for that assignment.

   To put it in simple words: if everybody is mortal then, in particular, John is mortal, and the mother of Mary is mortal, etc.
3. $\forall x(P(x) \to Q(x)), \forall x P(x) \models \forall x Q(x)$. *Why?*

   (Note that this is not *an instance of a propositional logical consequence.)*

4. $\exists x P(x) \wedge \exists x Q(x) \not\models \exists x(P(x) \wedge Q(x))$.

   Indeed, the structure $\mathcal{N}'$ obtained from $\mathcal{N}$ where $P(x)$ is interpreted as 'x is even' and $Q(x)$ is interpreted as 'x is odd' is a ***counter-model:***

   $\mathcal{N}' \models \exists x P(x) \wedge \exists x Q(x)$, *but* $\mathcal{N}' \not\models \exists x(P(x) \wedge Q(x))$.

Example 52 *Here we illustrate the 'semantic reasoning', based on the formal semantics of first-order formulae, for proving or disproving first-order logical consequences.*

1. *Show the logical validity of the following argument:*

   "If Charlie suffers from slyme disease and everyone who suffers from slyme disease has pink eyes, then somebody has pink eyes."

   Proof. *Let us first formalize the argument in first-order logic, by introducing predicates: $P(x)$ meaning 'x suffers from slyme disease', $Q(x)$ meaning 'x has pink eyes', and a constant symbol c interpreted as 'Charlie'.*

   Now the argument can be formalized as follows:

   (1) $(P(c) \wedge \forall x(P(x) \to Q(x)) \models \exists y Q(y)$.

   To show its validity take any structure $\mathcal{S}$ for the first-order language introduced above and any variable assignment v in $\mathcal{S}$. Now, suppose

   (2) $\mathcal{S}, v \models (P(c) \wedge \forall x(P(x) \to Q(x))$.

   We have to show that:

   (3) $\mathcal{S}, v \models \exists y Q(y)$.

   From the assumption (2) we have, by the truth definition of $\wedge$, that:

   (4) $\mathcal{S}, v \models P(c)$ *and*

   (5) $\mathcal{S}, v \models \forall x(P(x) \to Q(x))$.

   Let v' be a variable assignment obtained from v by re-defining it on x as follows: $v'(x) = c^{\mathcal{S}}$. (Recall, that $c^{\mathcal{S}}$ is the interpretation of the constant symbol c in $\mathcal{S}$) Then:

   (6) $\mathcal{S}, v' \models P(x)$.

   According to the truth definition of $\forall$, it follows from (5) that:

   (7) $\mathcal{S}, v' \models P(x) \to Q(x)$.

   From (6) and (7) it follows that:

   (8) $\mathcal{S}, v' \models Q(x)$.

   Now, let v'' be a variable assignment obtained from v by re-defining it on y as follows: $v''(y) = v'(x) = c^{\mathcal{S}}$. Then we have:

   (9) $\mathcal{S}, v'' \models Q(y)$.

   According to the truth definition of $\exists$, it follows from (9) that:

   (3) $\mathcal{S}, v \models \exists y Q(y)$, *so we are done.* ■

2. *Prove the logical non-validity of the following argument:*

   "If everything is black or white then everything is black or everything is white."

   Proof. *Again, we first formalize the argument in first-order logic, by introducing predicates:*

   $B(x)$ meaning 'x is black' and $W(x)$ meaning 'x is white'.

   Now the argument can be formalized as follows:

$$\forall x(B(x) \lor W(x)) \models \forall xB(x) \lor \forall xW(x)$$

   To falsify this logical consequence it suffices to find any counter-model*, i.e., a structure $\mathcal{S}$ and a variable assignment v such that*

   $\mathcal{S}, v \models \forall x(B(x) \lor W(x))$ and $\mathcal{S}, v \not\models \forall xB(x) \lor \forall xW(x)$.

   Note that the interpretations of the predicates B and W need not have anything to do with black, white, or any colours, in that structure. So, we choose to take the structure $\mathcal{S}$ with domain the set $\mathbf{Z}$ of all integers, where the interpretation of B is the predicate Even*, where* Even(n) *means 'n is even', and the interpretation of W is the predicate* Odd*, where* Odd(n) *means 'n is odd'.*

   We already know that the variable assignment is irrelevant here, because there are no free variables in this argument, but we will nevertheless need it in order to process the truth definitions of the quantifiers.

   So, consider any *variable assignment v in $\mathcal{S}$. Then take any variable assignment v' that differs from v possibly only on x. The integer $v'(x)$ is even or odd, therefore $\mathcal{S}, v' \models B(x)$ or $\mathcal{S}, v' \models W(x)$, hence (by the truth definition of $\lor$):*

   (1) $\mathcal{S}, v' \models B(x) \lor W(x)$.

   Thus, by the truth definition of $\forall$, it follows that:

   (2) $\mathcal{S}, v \models \forall x(B(x) \lor W(x))$

   On the other hand, consider the variable assignment v_1 obtained from v by re-defining it on x as follows: $v_1(x) = 1$. Then:

   (3) $\mathcal{S}, v_1 \not\models B(x)$.

   By the truth definition of $\forall$, it follows that:

   (4) $\mathcal{S}, v \not\models \forall xB(x)$.

   Likewise, consider the variable assignment v_2 obtained from v by re-defining it on x as follows: $v_2(x) = 2$. Then:

   (3) $\mathcal{S}, v_2 \not\models W(x)$.

   By the truth definition of $\forall$, it follows that:

   (4) $\mathcal{S}, v \not\models \forall xW(x)$.

   By the truth definition of $\lor$, it follows that

   (5) $\mathcal{S}, v \not\models \forall xB(x) \lor \forall xW(x)$.

Therefore, by the truth definition of $\to$, it follows from (2) and (5) that

(6) $\mathcal{S}, v \not\models \forall x(B(x) \vee W(x)) \to (\forall x B(x) \vee \forall x W(x))$. ■

Logical consequence in first-order logic satisfies all basic properties of propositional logical consequence. Besides, some important additional properties related to the quantifiers hold. They will be used as rules of inference in deductive systems for first-order logic, so we will give them more prominence.

Theorem 53 *For any first-order formulae $A_1, \dots, A_n, A, B$ the following hold:*

1. *If $A_1, \dots, A_n \models B$ then $\forall x A_1, \dots, \forall x A_n \models \forall x B$.*
2. *If $A_1, \dots, A_n \models B$ and $A_1, \dots, A_n$ are sentences, then $A_1, \dots, A_n \models \forall x B$, and hence $A_1, \dots, A_n \models \overline{B}$, where $\overline{B}$ is any universal closure of B.*
3. *If $A_1, \dots, A_n \models B[c/x]$ where c is a constant symbol not occurring in $A_1, \dots, A_n$, then $A_1, \dots, A_n \models \forall x B(x)$.*
4. *If $A_1, \dots, A_n, A[c/x] \models B$ where c is a constant symbol not occurring in $A_1, \dots, A_n, A$, or B, then $A_1, \dots, A_n, \exists x A \models B$.*
5. *For any term t free for substitution for x in A:*
   a) $\forall x A \models A[t/x]$.
   b) $A[t/x] \models \exists x A$.

2.4.4 Logical equivalence in first-order logic

Logical equivalence in first-order logic bears the same idea as in propositional logic: the first-order formulae A and B are logically equivalent if always one of them is true if and only if the other is true. Formally:

Definition 54 *The first-order formulae A and B are* ***logically equivalent****, denoted $A \equiv B$, if for every structure $\mathcal{S}$ and variable assignment v in $\mathcal{S}$:*

$$\mathcal{S}, v \models A \text{ if and only if } \mathcal{S}, v \models B.$$

In particular, if A and B are sentences, $A \equiv B$ means that every model of A is a model of B, and every model of B is a model of A.

The following theorem summarizes some basic properties of logical equivalence.

Theorem 55

1. $A \equiv A$
2. *If $A \equiv B$ then $B \equiv A$.*
3. *If $A \equiv B$ and $B \equiv C$ then $A \equiv C$.*
4. *If $A \equiv B$ then $\neg A \equiv \neg B$, $\forall x A \equiv \forall x B$, and $\exists x A \equiv \exists x B$.*

5. *If* $A_1 \equiv B_1$ *and* $A_2 \equiv B_2$ *then* $A_1 \circ A_2 \equiv B_1 \circ B_2$ *where* $\circ$ *is any of* $\wedge, \vee, \rightarrow, \leftrightarrow$.
6. *The following are equivalent:*
   a) $A \equiv B$.
   b) $\models A \leftrightarrow B$.
   c) $A \models B$ *and* $B \models A$.

Thus, logical equivalence is reducible to logical consequence and to logical validity.

Theorem 56 *The result of renaming of any variable in any formula* A *is logically equivalent to* A. *Consequently, every formula can be transformed into a logically equivalent clean formula.*

Example 57

- *Any first-order instance of a pair of equivalent propositional formulae is a pair of logically equivalent formulae. For example:*

  $\neg\neg\exists x Q(x,y) \equiv \exists x Q(x,y)$ *(being an instance of* $\neg\neg p \equiv p$*);*

  $\exists x P(x) \rightarrow Q(x,y) \equiv \neg\exists x P(x) \vee Q(x,y)$ *(being an instance of* $p \rightarrow q \equiv \neg p \vee q$*).*
- $\exists x(5 < x \vee x^2 + x - \mathbf{2} = 0) \equiv \exists y(\mathbf{5} < y \vee y^2 + y - \mathbf{2} = \mathbf{0})$*, because the formula on the right is the result of renaming of the variable* x *in the formula on the left.*
- $\neg\exists x P(x) \not\equiv \exists x \neg P(x)$. *Indeed, for instance, 'There is no student who passed the exam' is not equivalent to 'There is a student who did not pass the exam'.*
- *"The integer* x *is not less than 0" is* not logically equivalent *to "The integer* x *is greater than or equal to 0". Mathematically these mean the same (due of the mathematical property of the ordering of integers called '*trichotomy*', which arranges all integers in a line) but* not *because of logical reasons. Likewise, 2+2=4 is a mathematical, not a logical truth.*

  To put it simply: Logic does not know any mathematics. *It is important to distinguish logical from non-logical truths, and logical from non-logical equivalences.*

2.4.5 Some important logical equivalences involving quantifiers

1. To begin with, the negation swaps the quantifiers as follows:

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

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

   For example:

   - "Not every student wrote the test" means the same as " There is a student who did not write the test"
   - "There is no natural number less than 0" means " Every natural number is not less than 0".

2. By negating both sides of the equivalences above we find that each of the universal and existential quantifier is *definable* in terms of the other:

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

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

3. Two nested quantifiers of the same type commute:

$$\forall x \forall y A \equiv \forall y \forall x A$$

$$\exists x \exists y A \equiv \exists y \exists x A$$

4. Consequently, every two universal closures of any given formula are logically equivalent.

5. The case of two different nested quantifiers is more delicate: while

$$\exists x \forall y A \rightarrow \forall y \exists x A$$

is valid (check that yourself!), the converse implication

$$\forall y \exists x A \rightarrow \exists x \forall y A$$

is not. For instance,

"For every integer x there is an integer y such that $x + y = 0''$,

which is true, certainly does not imply that

"There is an integer y such that for every integer x it holds that $x + y = 0.''$

which is false.

Likewise, 'Every man loves a woman' does not imply that 'There is a woman whom every man loves'.

Therefore, $\exists x \forall y A$ and $\forall y \exists x A$ are *not equivalent.* It is important to remember which of these imply the other, and why.

2.4.6 Exercises

1. Use semantic reasoning to decide which of the following first-order formulae are logically valid. For each of those which are not, give a *counter-model*, i.e. a structure which falsifies it.

   a) $\forall x(P(x) \vee \neg P(x))$;
   b) $\forall x P(x) \vee \forall x \neg P(x)$;
   c) $\forall x P(x) \vee \neg \forall x P(x)$;
   d) $\exists x(P(x) \vee \exists x \neg P(x))$;
   e) $\exists x P(x) \rightarrow \forall x P(x)$;

f) $\exists x P(x) \to \exists y P(y)$;

g) $\exists x(P(x) \to \forall y P(y))$;

h) $\exists x(P(x) \to \forall x P(x))$;

i) $\forall x(P(x) \to \exists y P(y))$;

j) $\forall x(P(x) \to \forall y P(y))$;

k) $\forall x \exists y Q(x, y) \to \forall y \exists x Q(x, y)$;

l) $\forall x \exists y Q(x, y) \to \forall y \exists x Q(y, x)$;

m) $\forall x(\exists y Q(x, y) \to \exists y Q(y, x))$;

n) $\exists x \neg \exists y P(x, y) \to \forall y \neg \forall x P(x, y)$;

o) $(\forall x \exists y P(x, y) \wedge \forall x \forall y(P(x, y) \to P(y, x))) \to \exists x P(x, x)$.

2. Using semantic arguments show that the following logical consequences hold.

a) $\forall x A(x) \models \neg \exists x \neg A(x)$;

b) $\neg \exists x \neg A(x) \models \forall x A(x)$;

c) $\exists x A(x) \models \neg \forall x \neg A(x)$;

d) $\neg \forall x \neg A(x) \models \exists x A(x)$;

e) $\exists x \exists y A(x, y) \models \exists y \exists x A(x, y)$.

3. Suppose x is not free in Q. Show that the following logical consequences hold by giving semantic arguments.

a) $\forall x(P(x) \vee Q) \models \forall x P(x) \vee Q$;

b) $\forall x P(x) \vee Q \models \forall x(P(x) \vee Q)$;

c) $\exists x(P(x) \wedge Q) \models \exists x P(x) \wedge Q$;

d) $\exists x P(x) \wedge Q \models \exists x(P(x) \wedge Q)$;

e) $\forall x(Q \to P(x)), Q \models \forall x P(x)$;

f) $\exists x P(x) \to Q \models \forall x(P(x) \to Q)$;

g) $\exists x(P(x) \to Q), \forall x P(x) \models Q$;

h) $\exists x(Q \to P(x)), Q \models \exists x P(x)$.

4. Using semantic reasoning, determine which of the following logical consequences hold. For those which do, try to give a semantic argument. For those which do not, construct a *counter-model*, i.e. a structure in which all premises are true, while the conclusion is false.

a) $\forall x A(x), \forall x B(x) \models \forall x(A(x) \wedge B(x))$;

b) $\forall x(A(x) \wedge B(x)) \models \forall x A(x) \wedge \forall x B(x)$;

c) $\forall x A(x) \vee \forall x B(x) \models \forall x(A(x) \vee B(x))$;

d) $\forall x(A(x) \vee B(x)) \models \forall x A(x) \vee \forall x B(x)$;

e) $\forall x A(x) \to \forall x B(x) \models \forall x (A(x) \to B(x))$;

f) $\forall x (A(x) \to B(x)) \models \forall x A(x) \to \forall x B(x)$;

g) $\exists x (A(x) \land B(x)) \models \exists x A(x) \land \exists x B(x)$;

h) $\exists x A(x), \exists x B(x) \models \exists x (A(x) \land B(x))$;

i) $\exists x (A(x) \lor B(x)) \models \exists x A(x) \lor \exists x B(x)$;

j) $\exists x A(x) \lor \exists x B(x) \models \exists x (A(x) \lor B(x))$;

k) $\exists x (A(x) \to B(x)) \models \exists x A(x) \to \exists x B(x)$.

l) $\exists x A(x) \to \exists x B(x) \models \exists x (A(x) \to B(x))$.

5. If x does not occur free in Q show that:

   a) $\forall x (P \lor Q) \equiv \forall x P \lor Q$;

   b) $\exists x (P \land Q) \equiv \exists x P \land Q$;

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

   d) $\forall x (P \to Q) \equiv \exists x P \to Q$;

   e) $\exists x (P \to Q) \equiv \forall x P \to Q$;

   f) $\exists x (Q \to P) \equiv Q \to \exists x P$.

6. Use semantic reasoning to decide which of the following logical equivalences hold. For each of those that do not hold, give a counter-model, i.e. a structure in which one formula is true while the other false.

   a) $\forall x (P(x) \land Q(x)) \equiv \forall x P(x) \land \forall x Q(x)$;

   b) $\forall x (P(x) \lor Q(x)) \equiv \forall x P(x) \lor \forall x Q(x)$;

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

   d) $\exists x (P(x) \land Q(x)) \equiv \exists x P(x) \land \exists x Q(x)$;

   e) $\exists x (P(x) \lor Q(x)) \equiv \exists x P(x) \lor \exists x Q(x)$;

   f) $\exists x (P(x) \to Q(x)) \equiv \exists x P(x) \to \exists x Q(x)$;

   g) $\forall x Q(x,x) \equiv \forall x \exists y Q(x,y)$;

   h) $\forall x \exists y Q(x,y) \equiv \exists x Q(x,x)$;

   i) $\exists x \forall y Q(x,y) \equiv \exists y Q(y,y)$;

   j) $\forall x \forall y Q(x,y) \equiv \forall x \forall y Q(y,x)$;

   k) $\exists x P(x) \to \forall x P(x) \equiv \exists x \neg P(x) \to \forall x \neg P(x)$;

7. Check for each of the following statements whether it logically implies the other, by formalizing them in first-order logic and using semantic arguments.

   A: “Not everybody who is a lawyer is greedy.”

   B: “There exists a lawyer and not everybody is greedy.”

If any of the logical consequences does not hold, give an appropriate counter-model.

8. Check for each of the following statements if it logically implies the other, by formalizing them in first-order logic and using semantic arguments.

   A: "For no object it is the case that if it is a plonk then it is a qlink."

   B: "There exists a plonk and there exists no qlink."

   If any of the logical consequences does not hold, give an appropriate counter-model.

3 Deduction in first-order logic

Validity, logical consequence and equivalence in first-order logic can be established by using deductive systems, like in propositional logic.

Here we will construct and demonstrate extensions of the deductive systems for propositional logic introduced earlier to first-order logic by adding additional axioms (for Axiomatic Systems), inference rules (Propositional Semantic Tableaux and Natural Deduction) or additional procedures handling for the quantifiers (for Resolution), that are sound and complete for first-order logic.

That sound and complete deductive systems for first-order logic exist is not obvious at all. Such (axiomatic) deductive system was first established by Kurt Gödel in 1929, in his doctoral thesis.

Unlike the propositional case, however, *none of these deductive systems can be guaranteed to terminate its search for a derivation*, even if such a derivation exists. This happens, for instance, when the input formula is not valid, but can only be falsified in an infinite counter-model.

In fact, it was proved by Alonso Church in 1936 that the problem whether a given first-order sentence is valid is *not algorithmically decidable.* Consequently, logical consequence in first-order logic is algorithmically undecidable, too.

Therefore, no sound, complete, and always terminating deductive system for first-order logic can be designed.

3.1 Semantic Tableaux for first-order logic

Here we extend the method of propositional Semantic Tableaux to first-order logic by adding rules for the quantifiers.

Quantifier rules for Semantic Tableaux

(∀T)
$$\begin{array}{c} \forall x A(x) : \mathtt{T} \\ \downarrow \\ A(t/x) : \mathtt{T} \end{array}$$
for any term t occurring on this branch and free for x in A

(∃T)*
$$\begin{array}{c} \exists x A(x) : \mathtt{T} \\ \downarrow \\ A(c/x) : \mathtt{T} \end{array}$$
for a *new* constant symbol c not yet occurring on this branch.

(∃F)
$$\begin{array}{c} \exists x A(x) : \mathtt{F} \\ \downarrow \\ A(t/x) : \mathtt{F} \end{array}$$
for any term t occurring on this branch and free for x in A

(∀F)*
$$\begin{array}{c} \forall x A(x) : \mathtt{F} \\ \downarrow \\ A(c) : \mathtt{F} \end{array}$$
for a *new* constant symbol c not yet occurring on this branch.

(∗) *This rule may only be applied once for the given formula on each branch.*

The rules are quite natural and do not need much discussion; their correctness follows from some semantic properties of quantifiers that we listed in the previous chapter. So, let us explain the proviso (∗). It is only needed to prevent 'redundant' applications of the rules (∃T) and (∀F). Note that without that proviso each of these rules could be applied in principle infinitely many times – once for every constant symbol in the language that does not occur yet on the current branch. Intuitively, however, such applications should be redundant, because we only need *one* witness of the truth (respectively, of falsity) of the formula assumed to be true in the rule (∃T) (respectively, assumed to be false in (∀F)), *if such witness exists at all.* So, we only need to introduce *one name* for such a witness in any given branch of the tableau. Thus, if the assumption of the existence of such witness can lead to a contradiction and close the branch, then one application of the rule should suffice; otherwise no repeated applications of the rule can have an effect at all, but would make it impossible to declare the branch, and the tableau construction, saturated.

Now, the idea of derivation in first-order logic by means of the method of Semantic Tableaux is essentially the same as in propositional logic: in order to prove that $A_1, \ldots, A_n \models B$, we search systematically for a *counter-model*, i.e. a structure where $A_1, \ldots, A_n$ and $\neg B$ are satisfied simultaneously. Closed and open branches and tableaux are defined as in Propositional Semantic Tableaux, but with the more refined notion of saturation of first-order tableau, as discussed above. A closed tableau with the formulae $A_1, \ldots, A_n, \neg B$ at the root certifies that there cannot be such a counter-model, and that constitute a Semantic Tableaux derivation of the logical consequence $A_1, \ldots, A_n \models B$, which will be denoted by $A_1, \ldots, A_n \vdash_{\mathbf{ST}} B$. On the other hand, a tableau with an open and saturated branch contains the information needed for the construction of such a counter-model.

Theorem 58 *The system of Semantic Tableaux for first-order logic is sound and complete,* *i.e., for every first-order formulae $A_1, \ldots, A_n, C$:*

$$A_1, \ldots, A_n, \vdash_{ST} C \;\; \textit{iff} \;\; A_1, \ldots, A_n, \models C.$$

Unlike propositional logic, where every tableau either closes or saturates and terminates as open, now a third case is possible too, where the tableau neither closes, nor produces an open and saturated branch. This case will be illustrated in the last example below.

Lastly, note that the Semantic Tableaux system presented here does not involve rules for the equality, but such rules can be added to produce a sound and complete system for first-order logic with equality.

3.1.1 Some derivations in Semantic Tableaux

Example 59 *Using Semantic Tableaux, check if $\forall x(Q \to P(x)) \vDash Q \to \forall x P(x)$, where x does not occur free in Q.*

$$\begin{tikzcd}
 & \forall x(Q \to P(x)) : \mathtt{T}, Q \to \forall x P(x) : \mathtt{F} \arrow[d] & \\
 & \forall x(Q \to P(x)) : \mathtt{T}, Q : \mathtt{T}, \forall x P(x) : \mathtt{F} \arrow[d] & \\
 & P(c) : \mathtt{F} \arrow[d] & \\
 & Q \to P(c) : \mathtt{T} \arrow[dl] \arrow[dr] & \\
Q : \mathtt{F} & & P(c) : \mathtt{T} \\
\times & & \times
\end{tikzcd}$$

The tableau above closes, implying that $\forall x(Q \to P(x)) \vdash_{\mathbf{ST}} Q \to \forall x P(x)$, hence $\forall x(Q \to P(x)) \vDash Q \to \forall x P(x)$ holds.

Example 60 *Using Semantic Tableaux, check if $\exists x(A(x) \to B(x)), \exists x A(x) \models \exists x B(x)$.*

$$\begin{tikzcd}
 & \exists x(A(x) \to B(x)) : \mathtt{T}, \exists x A(x) : \mathtt{T}, \exists x B(x) : \mathtt{F} \arrow[d] & \\
 & A(c_1) \to B(c_1) : \mathtt{T} \arrow[d] & \\
 & A(c_2) : \mathtt{T} \arrow[d] & \\
 & B(c_1) : \mathtt{F} \arrow[d] & \\
 & B(c_2) : \mathtt{F} \arrow[dl] \arrow[dr] & \\
A(c_1) : \mathtt{F} & & B(c_1) : \mathtt{T} \\
\bigcirc & & \times
\end{tikzcd}$$

Note that, due to the proviso $(*)$, the left hand branch if this tableau cannot be extended any further and therefore remains open. Thus, the tableau does not close, so $\exists x(A(x) \rightarrow B(x)), \exists x A(x) \models \exists x B(x)$ cannot be derived in **ST**, hence this consequence does not hold.

Example 61 *Using Semantic Tableau, check if* $\forall x \exists y A(x,y) \models \exists x A(x, c_0)$

$$\begin{array}{c}
\forall x \exists y A(x,y) : \mathtt{T}, \exists x A(x, c_0) : \mathtt{F} \\
\downarrow \\
A(c_0, c_0) : \mathtt{F} \\
\downarrow \\
\exists y A(c_0, y) : \mathtt{T} \\
\downarrow \\
A(c_0, c_1) : \mathtt{T} \\
\downarrow \\
A(c_1, c_0) : \mathtt{F} \\
\downarrow \\
\exists y A(c_1, y) : \mathtt{T} \\
\downarrow \\
A(c_1, c_2) : \mathtt{T} \\
\downarrow \\
A(c_2, c_0) : \mathtt{F} \\
\vdots
\end{array}$$

It should now be evident that this tableau can be extended *forever* without either closing or saturating. That indicates that $\forall x \exists y A(x,y) \models \exists x A(x, c_0)$ cannot be derived in **ST**, hence this consequence does not hold, but this fact cannot be formally established at any finite stage of the extension of the tableau. Thus, *the construction of a first-order tableau does not always terminate.* As we mentioned at the beginning of this chapter, there is a good reason behind this shortcoming: it has been proved impossible to algorithmically determine whether a given first-order formula is logically valid. Consequently, it is impossible to algorithmically determine whether the extension of a given tableau for first-order logic will terminate or not.

3.1.2 Exercises

1. Prove the following logical validities and consequences, using Semantic Tableau.

   a) $\models \forall x P(x) \rightarrow \forall y P(y)$;

   b) $\models \exists x P(x) \rightarrow \exists y P(y)$;

   c) $\models \exists x (P(x) \rightarrow \forall y P(y))$;

   d) $\forall x A(x) \models \neg \exists x \neg A(x)$;

   e) $\neg \exists x \neg A(x) \models \forall x A(x)$;

   f) $\exists x A(x) \models \neg \forall x \neg A(x)$;

   g) $\neg \forall x \neg A(x) \models \exists x A(x)$;

h) $\exists x \exists y A(x,y) \models \exists y \exists x A(x,y)$.

2. Assuming that x is not free in Q, prove the following using Semantic Tableau.

   a) $\forall x(P(x) \vee Q) \models \forall x P(x) \vee Q$;

   b) $\forall x P(x) \vee Q \models \forall x(P(x) \vee Q)$;

   c) $\exists x(P(x) \wedge Q) \models \exists x P(x) \wedge Q$;

   d) $\exists x P(x) \wedge Q \models \exists x(P(x) \wedge Q)$;

   e) $\forall x(Q \rightarrow P(x)), Q \models \forall x P(x)$;

   f) $\exists x P(x) \rightarrow Q \models \forall x(P(x) \rightarrow Q)$;

   g) $\exists x(P(x) \rightarrow Q), \forall x P(x) \models Q$;

   h) $\exists x(Q \rightarrow P(x)), Q \models \exists x P(x)$.

3. Check, using Semantic Tableau, which of the following logical consequences hold. For those which do not, use the tableau to construct a counter-model, i.e. a structure and assignment in which all premises are true, while the conclusion is false.

   a) $\forall x A(x), \forall x B(x) \models \forall x(A(x) \wedge B(x))$;

   b) $\forall x(A(x) \wedge B(x)) \models \forall x A(x) \wedge \forall x B(x)$;

   c) $\forall x A(x) \vee \forall x B(x) \models \forall x(A(x) \vee B(x))$;

   d) $\forall x(A(x) \vee B(x)) \models \forall x A(x) \vee \forall x B(x)$;

   e) $\forall x A(x) \rightarrow \forall x B(x) \models \forall x(A(x) \rightarrow B(x))$;

   f) $\forall x(A(x) \rightarrow B(x)) \models \forall x A(x) \rightarrow \forall x B(x)$;

   g) $\exists x(A(x) \wedge B(x)) \models \exists x A(x) \wedge \exists x B(x)$;

   h) $\exists x A(x), \exists x B(x) \models \exists x(A(x) \wedge B(x))$;

   i) $\exists x(A(x) \vee B(x)) \models \exists x A(x) \vee \exists x B(x)$;

   j) $\exists x A(x) \vee \exists x B(x) \models \exists x(A(x) \vee B(x))$;

   k) $\exists x(A(x) \rightarrow B(x)) \models \exists x A(x) \rightarrow \exists x B(x)$.

   l) $\exists x A(x) \rightarrow \exists x B(x) \models \exists x(A(x) \rightarrow B(x))$.

4. Check for each of the following statements whether it logically implies the other, by formalizing them in first-order logic and using Semantic Tableau.

   A: "Not everybody who is a lawyer is greedy."
   B: "There exists a lawyer and not everybody is greedy."

   If any of the logical consequences does not hold, give an appropriate counter-model.

5. Check for each of the following statements if it logically implies the other, by formalizing them in first-order logic and using Semantic Tableau.

   A: "For no object it is the case that if it is a plonk then it is a qlink."
   B: "There exists a plonk and there exists no qlink."

   If any of the logical consequences does not hold, give an appropriate counter-model.

3.2 Axiomatic system for first-order logic

Now we extend the propositional axiomatic system **H** to first-order logic by adding axioms and rules for the quantifiers, where $\forall$ is regarded as the only quantifier in the language and $\exists$ is definable in terms of it.

Additional axioms:

(Ax∀1) $\forall x(A(x) \rightarrow B(x)) \rightarrow (\forall x A(x) \rightarrow \forall x B(x))$;

(Ax∀2) $\forall x A(x) \rightarrow A[t/x]$ where t is any term free for substitution for x in A.

(Ax∀3) $A \rightarrow \forall x A$ where x is not free in the formula A.

Additional rule:

We must also add the following rule of deduction, known as *Generalization*:

$$\frac{A}{\forall x A}$$

where A is any formula and x any variable.

Example 62 *Derive* $\forall x(Q \rightarrow P(x)), \exists x \neg P(x) \vdash_{\mathbf{H}} \neg Q$, *where x is not free in Q.*

Eliminating $\exists$, we are to derive $\forall x(Q \rightarrow P(x)), \neg\forall x\neg\neg P(x) \vdash_{\mathbf{H}} \neg Q$.

1. $\vdash_{\mathbf{H}} P(x) \rightarrow \neg\neg P(x)$ *derived in the Propositional **H**.*
2. $\vdash_{\mathbf{H}} \forall x(P(x) \rightarrow \neg\neg P(x))$ *from 1 and Generalization.*
3. $\forall x(P(x) \rightarrow \neg\neg P(x)) \vdash_{\mathbf{H}} \forall x P(x) \rightarrow \forall x\neg\neg P(x)$ *by Axiom (Ax∀1).*
4. $\vdash_{\mathbf{H}} \forall x P(x) \rightarrow \forall x\neg\neg x P(x)$ *by 2, 3, Deduction theorem, and Modus Ponens.*
5. $\forall x(Q \rightarrow P(x)) \vdash_{\mathbf{H}} \forall x Q \rightarrow \forall x P(x)$ *by Axiom (Ax∀1).*
6. $\forall x(Q \rightarrow P(x)), \forall x Q \vdash_{\mathbf{H}} \forall x P(x)$ *by 5 and Deduction Theorem.*
7. $Q \vdash_{\mathbf{H}} \forall x Q$ *by Axiom (Ax∀3).*
8. $\forall x(Q \rightarrow P(x)), Q \vdash_{\mathbf{H}} \forall x P(x)$ *by 6,7 and Deduction Theorem.*
9. $\vdash_{\mathbf{H}} \neg\forall x\neg\neg P(x) \rightarrow \neg\forall x P(x)$ *by 4 and contraposition (derived in the propositional **H**).*
10. $\forall x(Q \rightarrow P(x)), \neg\forall x P(x) \vdash_{\mathbf{H}} \neg Q$ *by 8 and contraposition.*
11. $\forall x(Q \rightarrow P(x)), \neg\forall x\neg\neg P(x) \vdash_{\mathbf{H}} \neg Q$ *by 9, 10, and Deduction Theorem.*

Axioms for the equality:

If the language contains equality (which is usually the case) then the following additional axioms for it are needed:

(Ax = 1) $x = x$

(Ax = 2) $x = y \rightarrow y = x$

(Ax = 3) $x = y \wedge y = z \rightarrow x = z$.

(Ax f) $x_1 = y_1 \wedge ... \wedge x_n = y_n \rightarrow f(x_1, ..., x_n) = f(y_1, ..., y_n)$ for every n-ary functional symbol p.

(Ax r) $x_1 = y_1 \wedge ... \wedge x_n = y_n \rightarrow (p(x_1, ..., x_n) \rightarrow p(y_1, ..., y_n))$ for every n-ary predicate symbol p.

Theorem 63 *The axiomatic system for first-order logic is sound and complete, i.e., for every first-order formulae $A_1, \ldots, A_n, C$:*

$$A_1, \ldots, A_n, \vdash_H C \quad \textit{iff} \quad A_1, \ldots, A_n, \models C.$$

3.2.1 Exercises

1. Prove the following logical validities and consequences, using the axiomatic system **H**.

   a) $\models \forall x P(x) \rightarrow \forall y P(y)$;

   b) $\models \exists x P(x) \rightarrow \exists y P(y)$;

   c) $\models \exists x (P(x) \rightarrow \forall y P(y))$;

   d) $\forall x A(x) \models \neg \exists x \neg A(x)$;

   e) $\neg \exists x \neg A(x) \models \forall x A(x)$;

   f) $\exists x A(x) \models \neg \forall x \neg A(x)$;

   g) $\neg \forall x \neg A(x) \models \exists x A(x)$;

   h) $\exists x \exists y A(x, y) \models \exists y \exists x A(x, y)$.

2. Suppose x is not free in Q. Prove the following logical consequences, using the axiomatic system **H**.

   a) $\forall x (P(x) \vee Q) \models \forall x P(x) \vee Q$;

   b) $\forall x P(x) \vee Q \models \forall x (P(x) \vee Q)$;

   c) $\exists x (P(x) \wedge Q) \models \exists x P(x) \wedge Q$;

   d) $\exists x P(x) \wedge Q \models \exists x (P(x) \wedge Q)$;

   e) $\forall x (Q \rightarrow P(x)), Q \models \forall x P(x)$;

   f) $\exists x P(x) \rightarrow Q \models \forall x (P(x) \rightarrow Q)$;

   g) $\exists x (P(x) \rightarrow Q), \forall x P(x) \models Q$;

   h) $\exists x (Q \rightarrow P(x)), Q \models \exists x P(x)$.

3.3 Natural Deduction for first-order logic

Now, we extend the propositional Natural Deduction to first-order logic by adding rules for the quantifiers.

3.3.1 Natural Deduction rules for the quantifiers

Introduction rules:

$$(\forall I)^{*} \quad \frac{A[c/x]}{\forall x A(x)}$$

$$(\exists I)^{**} \quad \frac{A[t/x]}{\exists x A(x)}$$

Elimination rules:

$$(\forall E)^{**} \quad \frac{\forall x A(x)}{A[t/x]}$$

$$(\exists E)^{***} \quad \frac{\exists x A(x) \qquad \begin{array}{c}[A[c/x]] \\ \vdots \\ C\end{array}}{C}$$

*where c is a constant symbol, not occurring in $A(x)$, nor in any open assumption used in the derivation of $A[c/x]$.

**for any term t free for x in A.

***where c is a constant symbol, not occurring in $A(x)$, nor in C or in any open assumption in the derivation of C, except for $A[c/x]$.

Let us first discuss the quantifier rules:

$(\forall I)$: If we are to prove a universally quantified sentence, $\forall x A(x)$, we reason as follows: we say 'Let c be any object from the domain (e.g. an arbitrary real number).' The name c of that arbitrary object must be a new one, however, which has not been used in the proof yet, to be sure that it is indeed arbitrary. Then we try to prove that $A(c)$ holds, without assuming any specific properties of c. If we succeed, then our proof will apply to *any* object c from the structure, so we will have a proof that $A(x)$ holds for *every* object x, i.e. a proof of $\forall x A(x)$.

$(\exists I)$: If we are to prove an existentially quantified sentence $\exists x A(x)$, then we try to find an explicit *example* c such that $A(c)$.

$(\forall E)$: If a premise is a universally quantified sentence $\forall x A(x)$, we can assume $A(a)$ for any object a from the structure. Within the formal system, instead of elements of a structure, we must use syntactic objects which represent them, viz. terms[1].

$(\exists E)$: If a premise is an existentially quantified sentence $\exists x A(x)$, then we introduce a name, say c, for an object that satisfies A, that is, we say, 'Well, if there is an x such that $A(x)$, then take one and call it c.' Then we replace the premise by $A[c/x]$. If we succeed in deriving the desired conclusion C from that new premise, then it can be derived from $\exists x A(x)$, provided c does not occur in A or C, nor in any other assumption

[1]Note that this sounds a little weaker, because in general not every element of the domain has a name, or is the value of a term. However, it is sufficient because ad hoc names can be added whenever necessary.

used in the derivation of C from $A[c/x]$, which means that the name c must be a new one, that has not been used in the proof yet.

Henceforth, we will write $A(t)$ instead of $A[t/x]$ whenever it is clear from the context which variable is substituted by the term t.

Remark A common mistake is to use the following simpler rule for elimination of $\exists$:

$$(\exists E) \qquad \frac{\exists x A(x)}{A(c)}$$

where c is a new constant symbol. Though simple and looking natural, this rule is *not valid*! Using it you can derive the invalid implication $\exists x A(x) \rightarrow A(c)$ which can be interpreted e.g. as "*If anyone will fail the exam then* $\langle$`YourName`$\rangle$ *will fail the exam*", which you certainly do not wish to be true. So, never use that simple 'rule'!

Theorem 64 *The system of Natural Deduction for first-order logic is sound and complete, i.e., for every first-order formulae $A_1, \ldots, A_n, C$:*

$$A_1, \ldots, A_n, \vdash_{ND} C \quad \textit{iff} \quad A_1, \ldots, A_n, \models C.$$

Lastly, note that the Natural Deduction system presented here does not involve rules for the equality, but such rules can be added to produce a sound and complete system for first-order logic with equality.

3.3.2 Derivations in first-order Natural Deduction

Derivations in first-order Natural Deduction are organized just like derivations in Propositional Natural Deduction: trees growing upwards (but constructed downwards), with nodes decorated by formulae, where the leaves are decorated with given or introduced assumptions, every node is branching upwards according to some of the derivation rules, where the formula decorating that node is the conclusion of the rule, while those decorating the successor nodes are the premises. The root of the derivation tree is decorated with the formula to be derived. In particular, the Natural Deduction derivation tree for the logical consequence $A_1, \ldots, A_n \models B$ has leaves decorated with formulae amongst $A_1, \ldots, A_n$ or with *subsequently cancelled* additional assumptions, while the root is decorated with B. If such derivation tree exists, then we say that $A_1, \ldots, A_n \models B$ is derivable in Natural Deduction and write $A_1, \ldots, A_n \vdash_{\mathbf{ND}} B$.

Here are some examples of derivations in Natural Deduction. Check that the rules for the quantifiers have been applied correctly.

1. $\vdash_{\mathbf{D}} \forall x\forall y P(x,y) \to \forall y\forall x P(x,y)$:

$$
{\scriptstyle(\to I)}\dfrac{{\scriptstyle(\forall I)}\dfrac{{\scriptstyle(\forall I)}\dfrac{{\scriptstyle(\forall E)}\dfrac{{\scriptstyle(\forall E)}\dfrac{[\forall x\forall y P(x,y)]^1}{\forall y P(c_1,y)}}{P(c_1,c_2)}}{\forall x P(x,c_2)}}{\forall y\forall x P(x,y)}}{\forall x\forall y P(x,y) \to \forall y\forall x P(x,y)}\,1
$$

2. $\forall x(P(x) \land Q(x)) \vdash_{\mathbf{D}} \forall x P(x) \land \forall x Q(x)$:

$$
{\scriptstyle(\land I)}\dfrac{{\scriptstyle(\forall I)}\dfrac{{\scriptstyle(\land E)}\dfrac{{\scriptstyle(\forall E)}\dfrac{\forall x(P(x) \land Q(x))}{P(c) \land Q(c)}}{P(c)}}{\forall x P(x)} \qquad {\scriptstyle(\forall I)}\dfrac{{\scriptstyle(\land E)}\dfrac{{\scriptstyle(\forall E)}\dfrac{\forall x(P(x) \land Q(x))}{P(c) \land Q(c)}}{Q(c)}}{\forall x Q(x)}}{\forall x P(x) \land \forall x Q(x)}
$$

3. $\neg\exists x\neg A(x) \vdash_{\mathbf{D}} \forall x A(x)$:

$$
\dfrac{\dfrac{\dfrac{\neg\exists x\neg A(x) \qquad \dfrac{[\neg A(c)]^1}{\exists x\neg A(x)}}{\bot}}{A(c)}\,1}{\forall x A(x)}
$$

4. $\neg\forall x\neg A(x) \vdash_{\mathbf{D}} \exists x A(x)$:

$$
\dfrac{\dfrac{\neg\forall x\neg A(x) \qquad \dfrac{\dfrac{\dfrac{[\neg\exists x A(x)]^2 \qquad \dfrac{[A(c)]^1}{\exists x A(x)}}{\bot}}{\neg A(c)}\,{\scriptstyle 1}}{\forall x\neg A(x)}}{\bot}}{\exists x A(x)}\,2
$$

5. Suppose x is not free in P. Then $\vdash_{\mathbf{D}} \forall x(P \to Q(x)) \to (P \to \forall x Q(x))$:

$$
{\scriptstyle(\to I)}\dfrac{{\scriptstyle(\to I)}\dfrac{{\scriptstyle(\forall I)}\dfrac{{\scriptstyle(\to E)}\dfrac{{\scriptstyle(\forall E)}\dfrac{[\forall x(P \to Q(x))]^1}{P \to Q(c)}, \qquad [P]^2}{Q(c)}}{\forall x Q(x)}}{P \to \forall x Q(x)}\,2}{\forall x(P \to Q(x)) \to (P \to \forall x Q(x))}\,1
$$

6. Suppose x is not free in P. Then $\forall x(Q(x) \to P), \exists x Q(x) \vdash_{\mathbf{D}} P$:

$$(\exists E)\,\dfrac{\exists x Q(x) \qquad (\to E)\,\dfrac{(\forall E)\,\dfrac{\forall x(Q(x) \to P)}{Q(c) \to P}, \qquad [Q(c)]^1}{P}}{P}\,1$$

3.3.3 Exercises

In all Natural Deduction derivations, indicate the succession of all steps by numbering them, and check that the provisos for all applications of rules are satisfied.

1. Prove the following logical validities and consequences, using Natural Deduction.

   a) $\models \forall x P(x) \to \forall y P(y)$;

   b) $\models \exists x P(x) \to \exists y P(y)$;

   c) $\models \exists x(P(x) \to \forall y P(y))$;

   d) $\forall x A(x) \models \neg\exists x \neg A(x)$;

   e) $\neg\exists x \neg A(x) \models \forall x A(x)$;

   f) $\exists x A(x) \models \neg\forall x \neg A(x)$;

   g) $\neg\forall x \neg A(x) \models \exists x A(x)$;

   h) $\exists x \exists y A(x, y) \models \exists y \exists x A(x, y)$.

2. Suppose x is not free in Q. Prove the following logical consequences, using Natural Deduction.

   a) $\forall x(P(x) \vee Q) \models \forall x P(x) \vee Q$;

   b) $\forall x P(x) \vee Q \models \forall x(P(x) \vee Q)$;

   c) $\exists x(P(x) \wedge Q) \models \exists x P(x) \wedge Q$;

   d) $\exists x P(x) \wedge Q \models \exists x(P(x) \wedge Q)$;

   e) $\forall x(Q \to P(x)), Q \models \forall x P(x)$;

   f) $\exists x P(x) \to Q \models \forall x(P(x) \to Q)$;

   g) $\exists x(P(x) \to Q), \forall x P(x) \models Q$;

   h) $\exists x(Q \to P(x)), Q \models \exists x P(x)$;

   i) $\exists x(\neg P(x) \vee Q) \models \forall x P(x) \to Q$.

   j) $\exists x(P(x) \vee \neg Q) \models Q \to \exists x P(x)$.

3. Determine which of the following first-order logical consequences hold by looking for a derivation in ND. For those that you cannot find such derivation, consider A and B as unary predicates and look for a counter-model, i.e. a structure and assignment in which all premises are true, while the conclusion is false.

a) $\forall x A(x), \forall x B(x) \models \forall x(A(x) \wedge B(x))$;

b) $\forall x(A(x) \wedge B(x)) \models \forall x A(x) \wedge \forall x B(x)$;

c) $\forall x A(x) \vee \forall x B(x) \models \forall x(A(x) \vee B(x))$;

d) $\forall x(A(x) \vee B(x)) \models \forall x A(x) \vee \forall x B(x)$;

e) $\forall x A(x) \rightarrow \forall x B(x) \models \forall x(A(x) \rightarrow B(x))$;

f) $\forall x(A(x) \rightarrow B(x)) \models \forall x A(x) \rightarrow \forall x B(x)$;

g) $\exists x(A(x) \wedge B(x)) \models \exists x A(x) \wedge \exists x B(x)$;

h) $\exists x A(x), \exists x B(x) \models \exists x(A(x) \wedge B(x))$;

i) $\exists x(A(x) \vee B(x)) \models \exists x A(x) \vee \exists x B(x)$;

j) $\exists x A(x) \vee \exists x B(x) \models \exists x(A(x) \vee B(x))$;

k) $\exists x(A(x) \rightarrow B(x)) \models \exists x A(x) \rightarrow \exists x B(x)$;

l) $\exists x A(x) \rightarrow \exists x B(x) \models \exists x(A(x) \rightarrow B(x))$.

3.4 Prenex and clausal normal forms

3.4.1 Negating first-order formulae. Negation normal form.

Using the first pair of equivalences in section 2.4.5, in addition to those from propositional logic, we can now systematically negate not only propositions, but any first-order sentences, as well and transform them into *negation normal form*, where negation only occurs in front of atomic formulae. For example, the sentence

"For every car, there is a driver who, if (s)he can start it, then (s)he can stop it."

can be formalized in a suitable language as

$$\forall x(\mathsf{Car}(x) \to \exists y(\mathsf{Driver}(y) \land (\mathsf{Start}(x,y) \to \mathsf{Stop}(x,y)))).$$

Now, we negate:

$$\begin{aligned}
&\neg\forall x(\mathsf{Car}(x) \to \exists y(\mathsf{Driver}(y) \land (\mathsf{Start}(x,y) \to \mathsf{Stop}(x,y))))\\
\equiv\ &\exists x\neg(\mathsf{Car}(x) \to \exists y(\mathsf{Driver}(y) \land (\mathsf{Start}(x,y) \to \mathsf{Stop}(x,y))))\\
\equiv\ &\exists x(\mathsf{Car}(x) \land \neg\exists y(\mathsf{Driver}(y) \land (\mathsf{Start}(x,y) \to \mathsf{Stop}(x,y))))\\
\equiv\ &\exists x(\mathsf{Car}(x) \land \forall y\neg(\mathsf{Driver}(y) \land (\mathsf{Start}(x,y) \to \mathsf{Stop}(x,y))))\\
\equiv\ &\exists x(\mathsf{Car}(x) \land \forall y(\neg\mathsf{Driver}(y) \lor \neg(\mathsf{Start}(x,y) \to \mathsf{Stop}(x,y))))\\
\equiv\ &\exists x(\mathsf{Car}(x) \land \forall y(\neg\mathsf{Driver}(y) \lor (\mathsf{Start}(x,y) \land \neg\mathsf{Stop}(x,y)))).
\end{aligned}$$

Since $\neg A \lor B \equiv A \to B$, the last formula is equivalent to

$$\exists x(\mathsf{Car}(x) \land \forall y(\mathsf{Driver}(y) \to (\mathsf{Start}(x,y) \land \neg\mathsf{Stop}(x,y)))).$$

Thus, the negation of the sentence above is equivalent to:

"There is a car such that every driver can start it and cannot stop it."

Note that the restricted quantifiers satisfy the same equivalences (and non-equivalences) mentioned in 2.4.5. In particular,

$$\begin{aligned}
\neg\forall x(P(x) \to A) &\quad\equiv\quad \exists x(P(x) \land \neg A)\\
\neg\exists x(P(x) \land A) &\quad\equiv\quad \forall x(P(x) \to \neg A),
\end{aligned}$$

and hence:

$$\begin{aligned}
\neg\forall x \in \mathbf{X}(A) &\quad\equiv\quad \exists x \in \mathbf{X}(\neg A)\\
\neg\exists x \in \mathbf{X}(A) &\quad\equiv\quad \forall x \in \mathbf{X}(\neg A)
\end{aligned}$$

Thus, the negation of the definition of a limit of a function in calculus:

$$\neg(\forall\epsilon > \mathbf{0}\exists\delta > \mathbf{0}(\mathbf{0} < |x-c| < \delta \to |f(x)-L| < \epsilon))$$

is logically equivalent to

$$\exists\epsilon > \mathbf{0}\forall\delta > \mathbf{0}(\mathbf{0} < |x-c| < \delta \land \neg|f(x)-L| < \epsilon).$$

3.4.2 Prenex normal forms

Definition 65 *An open first-order formula is in a **disjunctive** (respectively, **conjunctive) normal form** if it is the result of uniform substitution of atomic formulae for propositional variables in a propositional formula in DNF (respectively, CNF).*

Definition 66 *A first-order formula $Q_1x_1...Q_nx_nA$, where $Q_1, ..., Q_n$ are quantifiers and A is an open formula, is said to be in a **prenex form**. The quantifier string $Q_1x_1...Q_nx_n$ is called the **prefix**, and the formula A — the **matrix** of the prenex form.*

*If A is in a DNF (respectively, CNF) then $Q_1x_1...Q_nx_nA$ is in a **prenex disjunctive** (respectively, **conjunctive) normal form.***

Example 67

1. *The formula*
$$\forall x \exists y(\neg x > \mathbf{0} \vee (y > \mathbf{0} \wedge \neg x = y^2))$$
is in a prenex DNF, but not a prenex CNF.
2. *The formula*
$$\forall x \exists y(x > \mathbf{0} \rightarrow (y > \mathbf{0} \wedge x = y^2))$$
is in a prenex form but neither in DNF nor in CNF.
3. *The formula*
$$\forall x(x > \mathbf{0} \rightarrow \exists y(y > \mathbf{0} \wedge x = y^2))$$
is not in a prenex form.

Theorem 68 *Every first-order formula is equivalent to a formula in a prenex disjunctive normal form (PDNF) and to a formula in a prenex conjunctive normal form (PCNF).*

Proof: Here is an algorithm for construction of these normal forms. Given any formula A :

1. Eliminate all occurrences of $\rightarrow$ and $\leftrightarrow$ as in the propositional case.
2. Import all negations inside all other logical connectives and transform the formula to negation normal form.
3. Pull all quantifiers in front and thus transform the formula into a prenex form. For that use the equivalences

   a) $\forall xP \wedge \forall xQ \equiv \forall x(P \wedge Q)$,

   b) $\exists xP \vee \exists xQ \equiv \exists x(P \vee Q)$,

   to pull some quantifiers outwards and, after renaming the formula *wherever necessary.*

   Then, use also the following equivalences, where x does not occur free in Q, until the formula is transformed to a prenex form:

   c) $\forall xP \wedge Q \equiv Q \wedge \forall xP \equiv \forall x(P \wedge Q)$,

   d) $\forall xP \vee Q \equiv Q \vee \forall xP \equiv \forall x(P \vee Q)$,

   e) $\exists xP \vee Q \equiv Q \vee \exists xP \equiv \exists x(P \vee Q)$,

f) $\exists x P \wedge Q \equiv Q \wedge \exists x P \equiv \exists x(P \wedge Q)$,

4. Finally, transform the matrix in a DNF or CNF, just like a propositional formula.

Example 69 *Let* $A = \exists z(\exists x Q(x,z) \vee \exists x P(x)) \rightarrow \neg(\neg\exists x P(x) \wedge \forall x \exists z Q(z,x))$.

1. *Eliminating* $\rightarrow$: $A \equiv \neg\exists z(\exists x Q(x,z) \vee \exists x P(x)) \vee \neg(\neg\exists x P(x) \wedge \forall x \exists z Q(z,x))$
2. *Importing the negation:* $A \equiv \forall z(\neg\exists x Q(x,z) \wedge \neg\exists x P(x)) \vee (\neg\neg\exists x P(x) \vee \neg\forall x \exists z Q(z,x))$
   $\equiv \forall z(\forall x \neg Q(x,z) \wedge \forall x \neg P(x)) \vee (\exists x P(x) \vee \exists x \forall z \neg Q(z,x))$.
3. *Using the equivalences (a) and (b):* $A \equiv \forall z \forall x(\neg Q(x,z) \wedge \neg P(x)) \vee \exists x(P(x) \vee \forall z \neg Q(z,x))$.
4. *Renaming:* $A \equiv \forall z \forall x(\neg Q(x,z) \wedge \neg P(x)) \vee \exists y(P(y) \vee \forall w \neg Q(w,y))$.
5. *Using the equivalences (c)-(f) and pulling the quantifiers in front:*
   $A \equiv \forall z \forall x \exists y \forall w((\neg Q(x,z) \wedge \neg P(x)) \vee P(y) \vee \neg Q(w,y))$.
6. *The resulting formula is in a prenex DNF. For a prenex CNF we have to distribute the* $\vee$ *over* $\wedge$*:*
   $A \equiv \forall z \forall x \exists y \forall w((\neg Q(x,z) \vee P(y) \vee \neg Q(w,y)) \wedge (\neg P(x) \vee P(y) \vee \neg Q(w,y)))$.

Note that we could have renamed the formula A into a clean formula right from the beginning. That, however, would have made the resulting prenex formula much larger.

3.4.3 Skolemization

Skolemization is a procedure (first developed by the Norwegian logician Thoralf Skolem) for systematic elimination of the existential quantifiers in a first-order formula in a prenex form by way of uniform replacement of all occurrences of existentially quantified individual variables with terms headed by new functional symbols, called *Skolem functions*. In the traditional version of Skolemization these Skolem functions take as arguments all variables (if any) which are bound by universal quantifiers in the scope of which the given existential quantifier sits. In particular, existentially quantified variables not in the scope of any universal quantifiers are replaced by constant symbols, called *Skolem constants*.

We will explain the Skolemization procedure by some examples:

Example 70

1. *The result of Skolemization of the formula*
   $$\exists x \forall y \forall z(P(x,y) \rightarrow Q(x,z))$$
   is
   $$\forall y \forall z(P(c,y) \rightarrow Q(c,z)),$$
   where c is a new constant symbol, called ***Skolem constant****.*
2. *More generally, the result of Skolemization of the formula*
   $$\exists x_1 \cdots \exists x_k \forall y_1 \cdots \forall y_n A(x_1, \ldots, x_k, y_1, \ldots, y_n)$$

is

$$\forall y_1 \cdots \forall y_n A(c_1, \ldots, c_k, y_1, \ldots, y_n),$$

where $c_1, \ldots, c_k$ *are new Skolem constants.*

Note that the resulting formula is not equivalent *to the original one, but is* equally satisfiable with it.

3. *The result of Skolemization of the formula*

$$\exists x \forall y \exists z (P(x, y) \to Q(x, z))$$

is

$$\forall y (P(c, y) \to Q(c, f(y))),$$

where c *is a new Skolem constant and* f *is a new unary function, called* ***Skolem function.***

4. *More generally, the result of Skolemization of the formula*

$$\forall y \exists x_1 \cdots \exists x_k \forall y_1 \cdots \forall y_n A(y, x_1, \ldots, x_k, y_1, \ldots, y_n)$$

is

$$\forall y \forall y_1 \cdots \forall y_n A(y, f_1(y), \ldots, f_k(y), y_1, \ldots, y_n),$$

where $f_1, \ldots, f_k$ *are new Skolem functions.*

5. *The result of Skolemization of the formula*

$$\forall x \exists y \forall z \exists u A(x, y, z, u)$$

is the formula

$$\forall x \forall z A(x, f(x), z, g(x, z)),$$

where f *is a new unary Skolem function and* g *is a new binary Skolem function.*

Theorem 71 *The result of Skolemization of any first-order formula* A *is a formula which is generally not logically equivalent to* A*, but is equally satisfiable with* A*.*

The proof of this theorem is not difficult, but it requires use of a special set-theoretic principle called *the Axiom of Choice.*

Thus, for the purpose of checking (un)satisfiability of a formula, Skolemization is an admissible procedure.

3.4.4 Clausal forms

Recall, that a *literal* is an atomic formula or a negation of an atomic formula, and a *clause* is a set of literals (representing their disjunction).

Example of a clause:

$$\{P(x),\ \neg P(f(c, g(y))),\ \neg Q(g(x), y),\ Q(f(x, g(c)), g(g(g(y))))\}.$$

Note that all variables in a clause are implicitly assumed to be universally quantified.

A *clausal form* is a set of clauses (representing their conjunction), for example:

$\{$
$\{P(x)\}$,
$\{\neg P(f(c)), \neg Q(g(x,x), y)\}$,
$\{\neg P(f(y)), P(f(c)), Q(y, f(x))\}$
$\}$.

Theorem 72 *Every first-order formula A can be transformed to a clausal form $\{C_1, \ldots, C_k\}$ such that A is equally satisfiable with the universal closure $\overline{(C_1 \wedge \cdots \wedge C_k)}$ of the conjunction of all clauses, where the clauses are considered as disjunctions.*

The algorithm transforming a formula into clausal form:

1. Transform A to a prenex CNF.
2. Skolemize all existential quantifiers.
3. Remove all universal quantifiers.
4. Write the matrix (which is in CNF) as a set of clauses.

Example 73 *Consider the formula*

$$A = \exists z(\exists x Q(x,z) \vee \exists x P(x)) \rightarrow \neg(\neg\exists x P(x) \wedge \forall x \exists z Q(z,x)).$$

from example 69.

1. *Transforming A to a prenex CNF:*

$$A \equiv \forall z \forall x \exists y \forall w((\neg Q(x,z) \vee P(y) \vee \neg Q(w,y)) \wedge (\neg P(x) \vee P(y) \vee \neg Q(w,y))).$$

2. *Skolemize all existential quantifiers:*

   Skolem(A) =

$$\forall z \forall x \forall w((\neg Q(x,z) \vee P(f(z,x)) \vee \neg Q(w, f(z,x))) \wedge (\neg P(x) \vee P(f(z,x)) \vee \neg Q(w, f(z,x)))).$$

3. *Remove all universal quantifiers and write the matrix as a set of clauses:*

$$\text{clausal}(A) = \{\neg Q(x,z), P(f(z,x)), \neg Q(w, f(z,x))\}, \{\neg P(x), P(f(z,x)), \neg Q(w, f(z,x))\}.$$

3.4.5 Exercises

1. Negate each of the following formulae and transform the result into a negation normal form.

   a) $\forall x(x = x^2 \rightarrow x > 0)$;

   b) $\forall x((x = x^2 \wedge x > 1) \rightarrow x^2 < 1)$;

c) $\forall x \exists y(x > y \to x > y^2)$;

d) $\forall x(x = 0 \lor \exists y \neg(xy = x))$;

e) $\forall x \exists y(\neg x = y \to x > y)$;

f) $\exists x \exists y(x > y \lor -x > y)$;

g) $\exists x(P(x) \to \forall y P(y))$;

h) $\forall x(P(x) \to Q(x)) \to (\forall x P(x) \to \forall x Q(x))$;

i) $(\exists x P(x) \to \exists x Q(x)) \to \exists x(P(x) \to Q(x))$;

2. Transform each of the following formulae into a prenex DNF and a prenex CNF. Then Skolemize the resulting formula and transform it into clausal form.

a) $\forall x((\exists y P(y) \land \forall z Q(z, x)) \to \exists y Q(x, y))$

b) $\exists z(\exists x Q(x, z) \to (\exists x P(x) \lor \neg \exists z P(z)))$

c) $\forall y \neg(P(y) \leftrightarrow \exists x Q(y, x))$

d) $\forall z(\exists y P(y) \leftrightarrow Q(z, y))$

e) $(\exists x P(x) \to \neg \exists x Q(x)) \to \forall x(P(x) \to \neg Q(x))$

f) $\forall x(\neg \forall y Q(x, y) \land P(z)) \to \exists z(\forall y Q(z, y) \land \neg P(x))$

g) $\neg \exists x(\forall y(\exists z Q(y, z) \leftrightarrow P(z)) \land \forall z R(x, y, z))$

h) $\neg(\forall y(\forall z Q(y, z) \to P(z)) \to \exists z(P(z) \land \forall x(Q(z, y) \to Q(x, z))))$

i) $\neg(\neg \exists z(P(z) \land \forall x(Q(z, y) \to Q(x, z))) \to \neg \forall y(\forall z Q(y, z) \to P(z)))$

j) $\neg(\forall y(\neg \exists z Q(y, z) \to P(z)) \to \exists z((P(z) \to Q(z, y)) \land \neg \exists x R(x, y, z)))$

3.5 Resolution in first-order logic

For supplementary readings to this section, see links on the course website.

3.5.1 Basic Resolution rule in first-order logic

The propositional resolution rule, extended to first-order logic reads:

$$\frac{C \vee Q(s_1, \ldots, s_n), \quad D \vee \neg Q(s_1, \ldots, s_n)}{C \vee D}$$

This rule, however, is not strong enough. Example: the clause set $\{\{P(x)\}, \{\neg P(f(y))\}\}$ is not satisfiable, as it corresponds to the unsatisfiable formula $\forall x \forall y (P(x) \wedge \neg P(f(y)))$. However, the resolution rule above cannot produce an empty clause, because it *cannot unify* the two clauses in order to resolve them.

So, we need a stronger derivation mechanism, that can handle cases like this. There are two natural solutions:

1. *Ground resolution*[2]: generate *sufficiently many ground instances* of every clause (over the so called *Herbrand universe* of the language) and then apply the standard Resolution rule above.

   In the example, ground resolution would generate the ground clauses $\{P(f(c))\}$ and $\{\neg P(f(c))\}$ for some constant symbol c.

   This method is sound and complete but inefficient, as it leads to the generation of too many unnecessary clauses. It will not be discussed further.

2. *Resolution with unification*: introduce a stronger Resolution rule, that first tries to match a pair of clauses by applying a suitable substitution that would enable resolving that pair.

   We will present this method in detail.

3.5.2 Substitutions of terms for variables, revisited

Recall that substitution of a term t for a variable x in a term s, denoted by $s[t/x]$, is the result of *simultaneous* replacements of all occurrences of x in s by t. For instance:

$$f(g(x), f(y,x))[g(a)/x] = f(g(g(a)), f(y, g(a))).$$

This can generalize to simultaneous substitutions of several terms for *different* variables, denoted $s[t_1/x_1, \ldots, t_k/x_k]$. For instance:

$$f(g(x), f(y,x))[g(y)/x, f(x,b)/y] = f(g(g(y)), f(f(x,b), g(y))).$$

Note that the condition of simultaneity is important: if we first substituted $g(y)$ for x and then $f(x,b)$ for y the result would have been different (and wrong).

[2]Introduced by John Alan Robinson in: "A Machine-Oriented Logic Based on the Resolution Principle", JACM, 1965.

One can apply a substitution to several terms simultaneously, e.g., when a substitution acts on an atomic formula $P(t_1, \dots, t_n)$. For instance:

$$P(x, f(x, y), g(y))[g(y)/x, a/y] = P(g(y), f(g(y), a), g(a)).$$

Given the substitution $\sigma = [t_1/x_1, \dots, t_k/x_k]$, the set of variables $\{x_1, \dots, x_k\}$ is called the *domain* of σ, denoted $\mathrm{dom}(\sigma)$.

Substitutions can be *composed* by consecutive application: the composition of substitutions τ and σ is the substitution $\tau\sigma$ obtained by applying τ followed by applying σ to the result, i.e., $t\tau\sigma = (t\tau)\sigma$ (note the order). For instance:

$$\begin{gathered} f(g(x), f(y, x))[f(x, y)/x][g(a)/x, x/y] = \\ f(g(f(x, y)), f(y, f(x, y)))[g(a)/x, x/y] = \\ f(g(f(g(a), x)), f(x, f(g(a), x))). \end{gathered}$$

The composition of two substitutions $\tau = [t_1/x_1, \dots, t_k/x_k]$ and σ can be computed as follows:

1. Extend $\mathrm{dom}(\tau)$ to cover $\mathrm{dom}(\sigma)$ by adding to τ substitutions $[x/x]$ for every variable x in $\mathrm{dom}(\sigma) \setminus \mathrm{dom}(\tau)$.
2. Apply the substitution σ simultaneously to all terms $[t_1, \dots, t_k]$ to obtain the substitution $[t_1\sigma/x_1, \dots, t_k\sigma/x_k]$.
3. Remove from the result all cases x_i/x_i, if any.

For instance:

$$[f(x, y)/x, x/y][y/x, a/y, g(y)/z] = [f(y, a)/x, y/y, g(y)/z] = [f(y, a)/x, g(y)/z].$$

3.5.3 Unification of terms

A substitution σ that makes two terms s and t identical is called a **unifier of s and t**. For example:

- the substitution $[f(y)/x]$ unifies the terms x and $f(y)$;
- the substitution $[f(c)/x, c/y, c/z]$ unifies the terms $g(x, f(f(z)))$ and $g(f(y), f(x))$.
- There is no unifier for the pair of terms $f(x)$ and $g(y)$, nor for the pair of terms $f(x)$ and x.

Two terms are **unifiable** if they have a unifier, otherwise they are **non-unifiable**.

A substitution σ that makes (the respective arguments of) two literals $Q(s_1, \dots, s_n)$ and $Q(t_1, \dots, t_n)$ identical is called a **unifier** of $Q(s_1, \dots, s_n)$ and $Q(t_1, \dots, t_n)$.

For example:

- $[f(y)/x]$ unifies $P(x)$ and $P(f(y))$;

- $[f(c)/x, c/y, c/z]$ unifies $Q(x, c, f(f(z)))$ and $Q(f(y), z, f(x))$.

  Indeed, the results of applying the substitution σ are:

  $Q(x, c, f(f(z)))\sigma = Q(f(c), c, f(f(c)))$,

  $Q(f(y), z, f(x))\sigma = Q(f(c), c, f(f(c)))$.

Two terms (resp., two literals) are **unifiable** if they have a unifier, otherwise they are **non-unifiable**.

A unifier τ of two terms (resp., literals) is **more general**[3] than a unifier ρ, if there is a substitution σ such that $\rho = \tau\sigma$.

For instance, $\rho = [c/x, f(c)/y]$ is a unifier of the literals $P(f(x))$ and $P(y)$, but $\tau = [f(x)/y]$ is a more general unifier, because $\rho = \tau\sigma$, where $\sigma = [c/x]$.

A unifier of two terms (resp., literals) is their **most general unifier** (MGU) if it is more general than any unifier of theirs.

A most general unifier (if exists) need not be unique. For instance, in the example above, τ is a most general unifier, as well as $[z/x, f(z)/y]$ for any variable z.

Eventually, we are interested in unifying literals, i.e., (negated) atomic formulae. Note that two atomic formulae $P(t_1, \ldots, t_m)$ and $Q(s_1, \ldots, s_n)$ cannot be unified unless $P = Q$ and respectively $m = n$. If that is the case, in order to unify the formulae we need to unify the respective pairs of arguments, i.e. we are looking for a (most general) unifier of the system

$$t_1 = s_1,$$
$$\vdots$$
$$t_n = s_n.$$

Informally, we do that by computing a most general unifier (if one exists) of the first pair, then applying it to both lists of terms, then computing a most general unifier of the next pair (if there is one) in the resulting lists, applying it to both resulting lists of terms, etc. In order to unify the current pair of terms we apply the algorithm recursively to the pair of lists of their respective arguments. The composition of all most general unifiers computed as above, if they all exist, is a most general unifier of the two input lists.

Here is a pseudocode of a simple recursive algorithm for computing a most general unifier of two lists of terms p and q:

```
procedure mgu(p, q)
  $\theta$ := $\varepsilon$   (the empty substitution).
  Scan p and q simultaneously, left-to-right,
    and search for the first corresponding subterms
    where p and q disagree (mismatch).
  If there is no mismatch, return $\theta$.
  Else, let s and t be the first respective subterms
```

[3]Note that every unifier is 'more general' than itself. A better, but more cumbersome terminology would be '*at least as general*'.

```
    in p and q where mismatch occurs.
  If variable(s) and s $\notin$ Var(t) then
    $\theta$ := compose($\theta$,[t/s]);
    $\theta$ := compose($\theta$, mgu(p$\theta$,q$\theta$)).
  Else, if variable(t) and t $\notin$ Var(s) then
    $\theta$ := compose($\theta$,[s/t]);
    $\theta$ := compose($\theta$, mgu(p$\theta$,q$\theta$)).
  Else, return failure.
end
```

Example 74 *Examples of unification of atomic formulae using the algorithm above:*

- *Literal 1:* `parents(x, father(x), mother(Bill))`,

  Literal 2: `parents(Bill, father(Bill), y)`,

  Most general unifier: `[Bill/x, mother(Bill)/y]`.
- *Literal 1:* `parents(x, father(x), mother(Bill))`,

  Literal 2: `parents(Bill, father(y), z)`,

  Most general unifier: `[Bill/x, Bill/y, mother(Bill)/z]`.
- *Literal 1:* `parents(x, father(x), mother(Jane))`,

  Literal 2: `parents(Bill, father(y), mother(y))`,

  *The procedure **mgu** starts computing: [* `Bill/x, Bill/y`*, ...failure].*

  The algorithm fails. Therefore, a unifier does not exist.
- *Term 1:* `g(x,f(x))`.

  Term 2: `g(f(y),y)`.

  *The procedure **mgu** starts computing: [*`f(y)/x`*,... failure].*

  MGU does not exist.

For more details and examples, see links on the course website.

3.5.4 Resolution with unification in first-order logic

The first-order resolution rule is combined with a preceding unification of the clausal set, in order to produce a complementary pair of literals in the resolving clauses:

In order not to weaken the resolvent unnecessarily, we require that the unifier applied in the rule is a most general unifier of the respective literals.

$$\textbf{Res}: \quad \frac{C \vee Q(s_1, \ldots, s_n), \quad D \vee \neg Q(t_1, \ldots, t_n)}{(C \vee D)\sigma}$$

where:

(i) The two clauses have no variables in common (achieved by renaming of variables),

(ii) σ is a most general unifier of the literals $Q(s_1, \ldots, s_n)$ and $Q(t_1, \ldots, t_n)$.

NB: we require that the unifier applied in the rule is a most general unifier of the respective literals in order not to weaken the resolvent unnecessarily.

Some examples of resolution with unification:

$$\frac{\{P(x)\}, \{\neg P(f(y))\}}{\{\}} \quad (MGU : [f(y)/x])$$

$$\frac{\{P(a,y),\ Q(y)\}\{\neg P(x,b), \neg Q(x)\}}{\{Q(b), \neg Q(a)\}} \quad (MGU : [a/x, b/y])$$

$$\frac{\{P(a,y),\ Q(y)\}\{\neg P(x,f(x)),\ Q(f(x))\}}{\{Q(f(a))\}} \quad (MGU : [a/x, f(a)/y])$$

Note that both literals in the resolvent become equal.

$$\frac{\{P(a,y),\ P(x,f(x))\}\{\neg P(x,f(a))\}}{\{\}} \quad (MGU : [a/x, f(a)/y]).$$

Note that, after unification, both literals in the first clause become $P(a, f(a))$.

Definition 75 *A* ***resolution-based derivation*** *of a formula C from a list of formulae $A_1, \ldots, A_n$, denoted $A_1, \ldots, A_n, \vdash_{Res} C$, is a derivation of the empty clause $\{\}$ from the set of clauses obtained from $A_1, \ldots, A_n, \neg C$, by successive applications of the rule* ***Res****.*

Theorem 76 *The system of Resolution for first-order logic is sound and complete, i.e. for every first-order formulae $A_1, \ldots, A_n, C$:*

$$A_1, \ldots, A_n, \vdash_{Res} C \ \ \textit{iff}\ \ A_1, \ldots, A_n, \models C.$$

The system of Resolution presented here does not involve rules for the equality, but such rules can be added to produce a sound and complete system for first-order logic with equality.

Unlike propositional Resolution, first-order Resolution *may run forever*, i.e., never terminate, in some cases when the conclusion does not follow logically from the premises.

It may also run forever even when the conclusion *does* follow logically from the premises, if unbounded number of unnecessary resolvents are produced.

To avoid that, special strategies or additional mechanisms for disposal of used-up clauses need to be applied.

3.5.5 Examples of resolution-based derivations

Example 77 *Prove using the method of first-order Resolution that:*

$$\forall x(P(x) \to Q(x)) \models \forall x P(x) \to \forall x Q(x).$$

1. *Transform*
$$\{\forall x(P(x) \to Q(x)), \neg(\forall x P(x) \to \forall x Q(x))\}$$
to clausal form:
$$\{\neg P(x), Q(x)\},\ \{P(y)\},\ \{\neg Q(c)\}$$
for some Skolem constant c.
2. *Successive applications of* ***Res****:*
   a) *Unify $P(x)$ and $P(y)$ with MGU $[y/x]$.*
   Then resolve $\{\neg P(x), Q(x)\}$ and $\{P(y)\}$ to obtain $\{Q(y)\}$.
   b) *Unify $Q(c)$ and $Q(y)$ with MGU $[c/y]$.*
   Then resolve $\{\neg Q(c)\}$ and $\{Q(y)\}$ to obtain $\{\}$.

Example 78 *Show that*

if Everybody loves somebody

and Everybody loves a lover

then Everybody loves everybody.

1. *Formalize the assumptions and the goal conclusion, using the predicate $L(x, y)$ meaning 'x loves y':*
   Everybody loves somebody: $\forall x \exists y L(x, y)$.
   Everybody loves a lover: $\forall x(\exists y L(x, y) \to \forall z L(z, x))$.
   Everybody loves everybody: $\forall x \forall y L(x, y)$.
2. *Transform the set*
$$\{\forall x \exists y L(x, y), \forall x(\exists y L(x, y) \to \forall z L(z, x)), \neg(\forall x \forall y L(x, y))\}$$
to a clausal form:
$$\{L(x, f(x))\},\ \{\neg L(x_1, y), L(z, x_1)\},\ \{\neg L(a, b)\}$$
for some Skolem constants a, b and a Skolem function f.
3. *Successive applications of* ***Res****:*
   a) *Unify $L(z, x_1)$ and $L(a, b)$ with MGU $[b/x_1, a/z]$ and resolve $\{\neg L(x_1, y), L(z, x_1)\}$ and $\{\neg L(a, b)\}$ to obtain $\{\neg L(b, y)\}$.*
   b) *Unify $L(x, f(x))$ and $L(b, y)$ with MGU $[b/x, f(b)/y]$ and resolve $\{L(x, f(x))\}$ and $\{\neg L(b, y)\}$ to obtain $\{\}$.*

Example 79 *Verify the logical consequence:*
$$\forall x \exists y R(x, y), \forall x \forall y \forall z((R(x, y) \wedge R(y, z)) \to R(x, z)) \models \exists x R(x, x).$$

1. *Transform*
$$\{\forall x \exists y R(x, y), \forall x \forall y \forall z((R(x, y) \wedge R(y, z)) \to R(x, z)), \neg(\exists x R(x, x))\}$$

to a clausal form:

$$C_1 = \{R(x, f(x))\}, C_2 = \{\neg R(u, y), \neg R(y, z), R(u, z)\}, C_3 = \{\neg R(v, v)\}$$

for some unary Skolem function f.

2. *Successive applications of* ***Res*** *to the clausal set*

$$C_1 = \{R(x, f(x))\}, C_2 = \{\neg R(u, y), \neg R(y, z), R(u, z)\}, C_3 = \{\neg R(v, v)\}$$

   a) *Unify $R(u, z)$ in C_2 and C_3, with MGU $[v/u, v/z]$ and resolve:*

      $C_4 = \{\neg R(v, y), \neg R(y, v)\}$.

   b) *Unify $R(v, y)$ in C_4 and C_1 with MGU $[w/x, w/y, f(w)/v]$ and resolve: $C_5 = \{\neg R(f(w), w)\}$.*

   c) *Unify $R(u, z)$ in C_2 and C_5 with MGU $[f(w)/u, w/z]$ and resolve: $C_6 = \{\neg R(f(w), y), \neg R(y, f(w))$*

   d) *Unify $R(f(w), y)$ in C_6 and C_1 with MGU $[f(w)/x, f(f(f(w))/y]$ and resolve:*

      $C_7 = \{\neg R(f(f(w)), f(w))\}$*, etc.*

Thus, an infinite set of clauses can be generated:

$$\{\neg R(f(w), w)\}, \{\neg R(f(f(w)), f(w))\}, \{\neg R(f(f(f(w))), f(f(w)))\}, \ldots$$

but the empty clause cannot be derived.

In fact, the logical consequence does not hold. A counter-model is the set of natural numbers with $R(x, y)$ interpreted as $x < y$.

3.5.6 Exercises

1. Prove the following logical validities and consequences, using the method of Resolution.

   a) $\models \forall x P(x) \to \forall y P(y)$;

   b) $\models \exists x P(x) \to \exists y P(y)$;

   c) $\models \exists x (P(x) \to \forall y P(y))$;

   d) $\forall x A(x) \models \neg \exists x \neg A(x)$;

   e) $\neg \exists x \neg A(x) \models \forall x A(x)$;

   f) $\exists x A(x) \models \neg \forall x \neg A(x)$;

   g) $\neg \forall x \neg A(x) \models \exists x A(x)$;

   h) $\exists x \exists y A(x, y) \models \exists y \exists x A(x, y)$.

2. Suppose x is not free in Q. Prove the following logical consequences, using the method of Resolution.

   a) $\forall x (P(x) \vee Q) \models \forall x P(x) \vee Q$;

   b) $\forall x P(x) \vee Q \models \forall x (P(x) \vee Q)$;

c) $\exists x(P(x) \wedge Q) \models \exists xP(x) \wedge Q$;

d) $\exists xP(x) \wedge Q \models \exists x(P(x) \wedge Q)$;

e) $\forall x(Q \to P(x)), Q \models \forall xP(x)$;

f) $\exists xP(x) \to Q \models \forall x(P(x) \to Q)$;

g) $\exists x(P(x) \to Q), \forall xP(x) \models Q$;

h) $\exists x(Q \to P(x)), Q \models \exists xP(x)$.

3. Using the method of Resolution, check which of the following logical consequences hold. For those which do not, construct a counter-model, i.e., a structure and assignment in which all premises are true, while the conclusion is false.

a) $\forall xA(x), \forall xB(x) \models \forall x(A(x) \wedge B(x))$;

b) $\forall x(A(x) \wedge B(x)) \models \forall xA(x) \wedge \forall xB(x)$;

c) $\forall xA(x) \vee \forall xB(x) \models \forall x(A(x) \vee B(x))$;

d) $\forall x(A(x) \vee B(x)) \models \forall xA(x) \vee \forall xB(x)$;

e) $\forall xA(x) \to \forall xB(x) \models \forall x(A(x) \to B(x))$;

f) $\forall x(A(x) \to B(x)) \models \forall xA(x) \to \forall xB(x)$;

g) $\exists x(A(x) \wedge B(x)) \models \exists xA(x) \wedge \exists xB(x)$;

h) $\exists xA(x), \exists xB(x) \models \exists x(A(x) \wedge B(x))$;

i) $\exists x(A(x) \vee B(x)) \models \exists xA(x) \vee \exists xB(x)$;

j) $\exists xA(x) \vee \exists xB(x) \models \exists x(A(x) \vee B(x))$;

k) $\exists x(A(x) \to B(x)) \models \exists xA(x) \to \exists xB(x)$;

l) $\exists xA(x) \to \exists xB(x) \models \exists x(A(x) \to B(x))$.

m) $\exists x(P(x) \vee Q(x)), \forall x(P(x) \to R(x)), \forall x(P(x) \to R(x)) \models \exists xR(x)$.

4. Formalize the following logical consequences in first-order logic and check for each of them whether it holds using Resolution.

a)

All logicians are clever.
All clever people are rich.
———
All logicians are rich.

b)

All natural numbers are integers.
Some integers are odd numbers.
———
Some natural numbers are odd numbers.

5. Check for each of the following statements whether it logically implies the other, by formalizing them in first-order logic and using Resolution.

A: "Not everybody who is a lawyer is greedy."

B: "There exists a lawyer and not everybody is greedy."

If any of the logical consequences does not hold, give an appropriate counter-model.

6. Check for each of the following statements if it logically implies the other, by formalizing them in first-order logic and using Resolution.

   A: "For no object it is the case that if it is a plonk then it is a qlink."

   B: "There exists a plonk and there exists no qlink."

   If any of the logical consequences does not hold, give an appropriate counter-model.

7. Using the predicates: $\mathbf{F}(\mathbf{x})$ for '$\mathbf{x}$ *is a fairy*', $\mathbf{E}(\mathbf{x})$ for '$\mathbf{x}$ *is an elf*', $\mathbf{G}(\mathbf{x})$ for '$\mathbf{x}$ *is a goblin*', $\mathbf{H}(\mathbf{x})$ for '$\mathbf{x}$ *is a hobbit*', $\mathbf{T}(\mathbf{x})$ for '$\mathbf{x}$ *is a troll*', $\mathbf{L}(\mathbf{x}, \mathbf{y})$ for '$\mathbf{x}$ *likes* $\mathbf{y}$', formalize the following logical consequence in first-order logic.

   "All fairies are elves.
   Hobbits do not like any elves.
   Kermit likes a goblin or a fairy.
   Everyone who likes any goblins likes no trolls.

   Therefore, if Kermit likes any trolls then Kermit is not a hobbit.

   Then check whether it is logically valid by applying Resolution.
   If it is not valid, give an appropriate counter-model.

8. Formalize the following argument in first-order logic by using suitable predicates, in the domain of all 'mythic creatures'.

   Then check if it is logically valid by applying Resolution. If it is not valid, give an appropriate counter-model.

   All trolls are elves or goblins.
   No troll can fly, unless it is an elf.
   All dragons can fly.
   All ugly goblins are trolls.
   Some goblins are ugly.

   Therefore, there is an ugly mythic creature that is not a dragon.

3.6 Addendum: some applications

3.6.1 Direct and indirect proofs

In mathematics we establish the truth of a statement by *proving* it. A *proof* may consist of a simple calculation (e.g. the proof of $12345679 \times 9 = 111111111$) or of a long and intricate argument involving a number of other, already proven statements, logical inferences, and possibly lots of complicated calculations, too. In this section we briefly discuss the *logical* aspects of proofs.

As we have already discussed, a typical (mathematical or not) statement looks like this:

If $\mathsf{P}_1, \ldots, \mathsf{P}_n$ then C.

where $P_1, ..., P_n$ are (possibly none) *premises* or *assumptions* and C is a *conclusion.*

The logical structure of a proof of such a statement naturally depends on the logical structure of the premises and the conclusion. Here we will provide some *proof tactics* telling us how to go about specific, *local* steps of the proof, but we will first discuss possible *proof strategies*, telling us how to organize proofs *globally.*

I. Direct proof. This proof strategy *assumes* that all premises are true, and then tries to *deduce* the desired conclusion by applying a sequence of correct (logical or substantial) inference steps. Schematically, a direct proof looks as follows:

$$\begin{gathered}\textsf{Assume } P_1, \ldots, P_n. \\ \vdots \\ \text{(a sequence of valid inferences)} \\ \vdots \\ \textsf{Conclude C.}\end{gathered}$$

Example 80 *A direct proof of the statement*

If n is an odd integer, then n^2 is an odd integer.

PROOF:

Suppose n is an odd integer.

Then there is an integer k such that $n = 2k + 1$.

Therefore, $n^2 = (2k + 1)(2k + 1) = 4k^2 + 4k + 1 = 2(2k^2 + 2k) + 1$.

Therefore, n^2 is odd.

II. Indirect proof, or **proof by contradiction**. The idea of this proof strategy is to assume that *all premises are true, while the conclusion is false* (hence *the negation of the conclusion is true*), and try to reach a *contradiction*, again by applying valid inferences only. We typically obtain a contradiction by deducing a statement known to be false (like deducing that $2 + 2 = 3$), or by deducing a statement and its negation (remember, a proposition $P \wedge \neg P$ is always false.) Often we reach a contradiction by deducing the negation of some of the premises (see the example below), which we have assumed to be true. The rationale behind the proof by contradiction is clear: if all our assumptions are true and we only apply valid inferences, then all our conclusions *must* be true, too. Therefore, by deducing a false conclusion, we show that, given that all original assumptions are true, the additional one that we made, namely the negation of the conclusion, must have been wrong, i.e. that *the conclusion must be true.*

Thus, the scheme of a proof by contradiction is:

Assume $P_1, \ldots, P_n$.

Assume $\neg C$

$\vdots$

(a sequence of valid inferences)

$\vdots$

Conclude A and $\neg A$

for some proposition A.

Example 81 *A proof by contradiction of the statement*

If x and y are integers, x is odd and xy is even, then y is even.

Proof:

Suppose x, y are integers, x is odd and xy is even.

Suppose y is not even.

Therefore, y is odd.

Then there is an integer k such that $x = 2k + 1$, since x is odd,

and there is an integer m such that $y = 2m + 1$, since y is odd,

Therefore, $xy = (2k + 1)(2m + 1) = \ldots = 2(2km + k + m) + 1$.

Therefore, xy is odd.

On the other hand, we have assumed that xy is even, and, therefore, xy is not odd.

Thus, xy is odd and xy is not odd.

This is a contradiction which completes the proof.

Example 82 *Euclid's proof by contradiction that*

$\sqrt{2}$ is irrational.

Proof:

Suppose it is not the case that $\sqrt{2}$ is irrational.

Therefore, $\sqrt{2}$ is rational.

Hence, $\sqrt{2}$ can be represented as an irreducible fraction, i.e. there are relatively prime integers m and n such that $\sqrt{2} = \frac{m}{n}$.

Therefore, $2 = \frac{m^2}{n^2}$, hence $2n^2 = m^2$.

Hence m^2 is even.

Therefore, m must be even, since if it were odd, then m^2 would have been odd

(see the example of a direct proof).

So, there is an integer a such that $m = 2a$.

Hence, $m^2 = 4a^2$.

Therefore, $2n^2 = 4a^2$, hence $n^2 = 2a^2$.

So, n^2 is even.

Therefore, n must be even, by the same argument as above.

Thus, there is an integer b such that $n = 2b$.

Therefore, the fraction $\frac{m}{n} = \frac{2a}{2b}$ is not irreducible,

which contradicts the assumption that $\frac{m}{n}$ is irreducible.

This contradiction completes the proof.

A particular case of proof by contradiction is **proof by contraposition.** This strategy is usually applied when the statement to be proved is of the type "If P, then C.", i.e. there is only one premise[4]. The proof by contraposition is based on the fact that the implication "If P, then C" is logically equivalent to its contrapositive "If $\neg C$, then $\neg P$." Therefore, if we prove the latter, we have a proof of the former. Usually we apply the proof by contraposition when it is easier to prove the contrapositive of the original implication.

Schematically, the proof by contraposition of "If P, then C" looks as follows:

Assume $\neg C$.

$\vdots$

(a sequence of valid inferences)

$\vdots$

Conclude $\neg P$.

As mentioned, a proof by contraposition is just a variant of a proof by contradiction: we only need to assume P together with $\neg C$ at the beginning, and then proving $\neg P$ will produce the desired contradiction.

Example 83 *A proof by contraposition of the statement*

[4]Though, it can be applied in the general case, since we can always replace all premises by their conjunction.

If n is an integer such that n^2 is even, then n is even.

PROOF:

Suppose n is not even.

Therefore, n is odd.

Then (see the example of a direct proof) n^2 is odd.

Therefore, n^2 is not even.

This completes the proof.

3.6.2 Tactics for logical reasoning

While Semantic Tableaux and Resolution are convenient and easier to use systems of deduction, they are only suitable for proofs by contradiction, but not for direct proofs. Here is some advice on *tactics* of proof, applicable to any of the methods discussed above. These tactics can be extracted from the rules of Natural Deduction. As mentioned earlier, the choice of tactics depends both on the logical structure of the conclusion and on the logical structure of the premises. We look at these separately.

Tactics based on the conclusion to be proved

- If the conclusion is a **negation**, $C = \neg A$, then we either import that negation inside, or (usually) apply proof by contradiction, viz. assume A (which is equivalent to the negation of $\neg A$) and try to reach a contradiction.
- If the conclusion is a **conjunction**, $C = A \wedge B$, we have to prove each of A and B.
- If the conclusion is a **disjunction**, $C = A \vee B$, we try to prove either of A or B. This works sometimes, but not always. If it does not work, then we can try a proof by contradiction, i.e. assume $\neg C$, which is equivalent to $\neg A \wedge \neg B$, hence we assume each of $\neg A$ and $\neg B$, and try to reach a contradiction. Finally, quite often we use the equivalence $P \vee Q \equiv \neg P \rightarrow Q$ and transform the disjunction into implication; then see below.
- If the conclusion is an **implication**, $C = A \rightarrow B$, we assume A in addition to all premises and try to prove B. Alternatively, we can attempt a proof by contraposition, by tackling $\neg B \rightarrow \neg A$ instead.
- If the conclusion is a **biconditional**, $C = A \leftrightarrow B$, then it is equivalent to the conjunction $(A \rightarrow B) \wedge (B \rightarrow A)$ and we follow the tactics mentioned above.
- If the conclusion is a **universally quantified sentence**, $C = \forall x A(x)$, we reason as follows: we say "Let c be an arbitrary object from the domain (e.g. an arbitrary real number)." The name c of that arbitrary object must be a new one, however, which has not been used in the proof yet. Then we try to prove that $A(c)$ holds, without assuming any specific properties of c. If we succeed, then our proof will apply to *any* object c from the domain, so we will have a proof that $A(x)$ holds for *every* object x, i.e. a proof of $\forall x A(x)$.

- If the conclusion is an **existentially quantified sentence**, $C = \exists x A(x)$, then we either try to find an explicit *example* c such that A(c) holds or we attempt a proof by double negation, i.e. a proof that *it cannot be the case that there does not exist* x *for which* $A(x)$ *holds.* The latter is called a *non-constructive proof of existence.* We shall come across such proofs in the course of calculus.

Tactics based on the premises

We use each premise P according to its logical form as follows:

- If the premise is a **negation**, $P = \neg A$, then we import that negation inside all other logical connectives.
- If the premise is a **conjunction**, $P = A \wedge B$, we replace it by two new premises, viz. A and B.
- If the premise is a **disjunction**, $P = A \vee B$, we reason *per cases*, i.e. we consider separately each of the two possible cases, viz. *Case 1:* A is true; *Case 2:* B is true. If we succeed to prove that in each of these cases the conclusion C follows, then this furnishes a proof that C follows from $A \vee B$. (Why?)
- If the premise is an **implication**, $P = A \to B$, we can replace it by the equivalent disjunction $\neg A \vee B$ and reason as in the previous case.
- If the premise is a **biconditional**, $P = A \leftrightarrow B$, then we replace it by the conjunction $(A \to B) \wedge (B \to A)$.
- If the premise is a **universally quantified sentence**, $P = \forall x A(x)$, we essentially replace it by new premises (possibly infinitely many) $A(c)$ for each object c from the domain.
- If the premise is an **existentially quantified sentence**, $P = \exists x A(x)$, then we introduce a name (say c) for an object that satisfies A, that is, we say, "Well, if there is an x such that $A(x)$, then take one and give it a name c." Then we replace the premise by $A(c)$. NB: the name c must be a new one that has not been used in the proof yet.

3.6.3 Automated reasoning and theorem proving

Reasoning is the process of deriving inferences from given facts or assumptions. Automated reasoning deals with the theory and computerized implementation of algorithms used to test if a given statement (the conjecture) is a logical consequence of a set of statements (axioms, facts, or assumptions).

A system of automated reasoning is based on a formal language for description of the axioms, facts, assumptions, and conjectures by means of formally constructed expressions in that language, generically known as *formulae.* A most common formal language for automated reasoning is first-order logic, and the most suitable and successful deductive systems used in automated theorem proving so far are resolution-based. The proofs produced by systems for automated reasoning describe how and why the conjecture follows from the axioms and hypotheses, in a manner that can be understood and agreed upon by everyone, and verified by means of a computer. The proof output may not only be a convincing argument that the conjecture is a logical consequence of the axioms and hypotheses, but it often also describes a process that may be implemented to solve some problems.

Systems for automated theorem proving are powerful computer programs, potentially capable of solving very difficult problems, practically impossible to handle by hand. Because of this extreme capability, their application and operation sometimes needs to be guided by an expert in the domain of application, in order to solve problems in a reasonable amount of time. Automated reasoning and theorem proving are particularly successful in many areas of mathematics, CS, and AI.

There are currently dozens of implemented automated theorem provers; see links to them on the course website.

A large repository of problems and benchmarks for automated theorem proving can be found on `http://www.cs.miami.edu/~tptp/index.html`.

For additional readings and further background on the topic see links on the course website.