Course Notes for MA4207 Mathematical Logic

Frank Stephan

March 26, 2018

These notes are closely following Chapters 0 – 2 of the textbook "A Mathematiucal Introduction to Logic" by Herbert B. Enderton. Therefore they are only for private viewing and should not be distributed over the internet; you can read them in IVLE. The lecture is based on these notes which I make for preparing the lecture. However, the lecture is not completely identical to the nodes and some examples might be given on the spot which are not in here. I will keep updating the notes on the way through the lecture.

Singapore, January - May 2018, Frank Stephan

Introduction.

Example. "All men are mortal. Sokrates is a man. Hence Socrates is mortal." Logic studies sentences and their consequences, in particular ways to deduce new knowledge from priorly known correct sentences.

Example. "All birds fly. Penguins are birds. Why do they not fly?" Assumption "All birds fly." is just wrong. From wrong assumptions, anything can be deducted. In logic, "All X do Y" should mean that every object in the set X does what the sentence Y says, while in everyday language, exceptions are often accepted.

General goals of the lecture.

1. What does it mean, the X follows logically from Y?
2. If a sentence X follows logically from a sentence Y, how does one prove this and is such a proof always possible?
3. Can everything what is correct about natural numbers also be proven? Or are there sentences X about natural numbers which are true but which cannot be proven? This will only be partially treated in this lecture and there is a more specialised lecture in Year 5 which gives a comprehensive answer to this question.
4. How do logic and recursion theory relate to each other.

First Model of Logic: Sentential Logic. This model is quite general and considers basic sentences $A_0, A_1, A_2, \ldots$ and then it studies what one can deduct about the truth of these sentences, when certain axioms are given, say "A_0 or A_1 is true" and "If A_2 is true, so are A_0 and A_1."

Second Model of Logic: First-Order Logic. The second model is taylormade for mathematics; it replaces atomic sentences by formal sentences about structures which can be expressed using quantifiers and variables and which are connected in the same way as atoms in sentential logic. This second model is called "First Order Logic". First Order Logic permits to quantify over members of sets; Second Order Logic permits to quantify over sets and functions. For this course, only Sentential

Logic and First Order Logic are studied. Example of First-Order Sentences over the natural numbers:

$$\begin{array}{lll} \forall x & \exists y & [x < y]; \\ \forall x & \forall y & [x + y = y + x]; \\ \exists x & \forall y & [x \leq y]; \\ \forall x & \forall y & [x \leq y \leftrightarrow \exists z\, [x + z = y]]. \end{array}$$

The first formula says that for every number x there is a number y which is strictly larger than x. The second formula says that the sum of two numbers does not depend on the order of these numbers. The third formula says that there is a natural number which is less or equal all other natural numbers; this number is 0. The fourth formula says that for all natural numbers x, y, the number x is less or equal to y iff y is the sum of x and some further natural number z.

Chapter 0 – Useful Facts About Sets.

Set theory codes up everything using sets, including natural numbers and real numbers, the latter are often identified with the subsets of the natural numbers or the functions from natural numbers to natural numbers. The coding will not be studied in this lecture.

Two sets X and Y are the same iff they have the same elements. The following operations on sets are quite common:

1. $X \mapsto X \cup \{t\}$: Adding the element t to X;
2. $Z = X \cup Y$: Z is the union of X and Y;
3. $Z = X \cap Y$: Z is the intersection of X and Y;
4. $Z = X - Y$: Z contains those elements of X which are not in Y;
5. $Z = \mathbb{P}(X)$: The elements of Z are all subsets Y of X;
6. $Z = \emptyset$: Z is the empty set which does not contain any element.

The subset-relation is defined as follows: $X \subseteq Y$ iff every element of X is also in Y; $X \subset Y$ iff $X \subseteq Y$ and $Y - X$ contains at least one element.

Special sets: $\emptyset$ is the empty set; $\mathbb{N}$ is the set of natural numbers, it is $\{0, 1, 2, 3, \ldots\}$. $\mathbb{Z}$ is the set of all integers, it is $\{\ldots, -3, -2, -1, 0, 1, 2, 3, \ldots\}$.

Selection of subsets: $\{t \in X : t \notin Y\}$ defines $X - Y$; one can also define informally the union as $\{t : t \in X \text{ or } t \in Y\}$; however, here the t is not bound to be in some set and then this definition sometimes leads to things which are not sets. The intersection is $\{t \in X : t \in Y\} = \{t : t \in X \text{ and } t \in Y\}$. General operators for sets:

$$\bigcup X = \{t : \exists Y \in X\, [t \in Y]\}$$

and

$$\bigcap X = \{t : \forall Y \in X\, [t \in Y]\}$$

and the latter is only a set when X is not empty.

Example. Consider $X = \{\{0, 1, 2\}, \{1, 2, 3\}, \{2, 3, 4\}\}$. Now $\bigcup X = \{0, 1, 2, 3, 4\}$ and $\bigcap X = \{2\}$.

Pairs and Sequences. An ordered pair $\langle X, Y\rangle$ can be formed by any definition which satisfies that $\langle X, Y\rangle = \langle V, W\rangle$ iff $X = V$ and $Y = W$. Furthermore, $\langle X, Y, Z\rangle$ is a short-hand for $\langle\langle X, Y\rangle, Z\rangle$. An example is to represent the set $\langle X, Y\rangle$ as $\{\{X, Y\}, \{Y\}\}$; however, any further definition which does this is okay. This definition generalises to $\langle x_1, x_2, \ldots, x_n, x_{n+1}\rangle = \langle\langle x_1, x_2, \ldots, x_n\rangle, x_{n+1}\rangle$. For example, $\langle 1, 2, 4, 8, 16\rangle$ stands for $\langle\langle\langle\langle 1, 2\rangle, 4\rangle, 8\rangle, 16\rangle$.

Lemma 0A. For all natural numbers n, m and all sequences $\langle x_1, \ldots, x_n\rangle$ and $\langle y_1, \ldots, y_{n+m}\rangle$, if $\langle x_1, \ldots, x_n\rangle = \langle y_1, \ldots, y_{n+m}\rangle$ then $x_1 = \langle y_1, \ldots, y_{m+1}\rangle$.

Proof. This one proves by induction over n. For $n = 1$, the equality follows from the way one writes the formulas, using that $\langle x_1\rangle = x_1$. If the Lemma 0A is true for a given n, then it also holds for $n + 1$, as $\langle x_1, \ldots, x_{n+1}\rangle = \langle\langle x_1, \ldots, x_n\rangle, x_{n+1}\rangle$ and $\langle y_1, \ldots, y_n, \ldots, y_{n+m+1}\rangle = \langle\langle y_1, \ldots, y_n, \ldots, y_{n+m}\rangle, y_{n+m+1}\rangle$ and from the equality of the two sequences follows the equality of their first components of the pairs equal to it and then the statement follows from induction hypothesis. Thus the inductive step goes through for all n and so the statement is true for all n by induction. □

Cartesian Products and Powers. The set $X \times Y$ is the set of all pairs $\langle a, b\rangle$ with $a \in X$ and $b \in Y$. Furthermore, the set Y^n is the set Y^n of sequences of length n of n elements $x_1, \ldots, x_n \in Y$ and the formal definition is the following one:

$$Y^n = \{\langle x_1, x_2, \ldots, x_n\rangle : x_1, x_2, \ldots, x_n \in Y\}.$$

For example, $Y^3 = Y \times Y \times Y$, where, more formally, one would write $(Y \times Y) \times Y$.

Relations. A relation is just a set of n-tuples where the m-th component has to come from the m-th base set. In particular one is interested in relations between two sets X and Y. For such an $R \subseteq X \times Y$, one denotes with $dom(R)$ the domain $\{x : \exists y \in Y : \langle x, y\rangle \in R\}$ and $ran(R)$ the range $\{y : \exists x \in X : \langle x, y\rangle \in R\}$.

A binary relatin R is reflexive on X iff $\langle y, y\rangle \in R$ for all $y \in R$.
R is symnmetric iff $\langle x, y\rangle \in R$ always implies $\langle y, x\rangle \in R$.
R is transitive iff the statements $\langle x, y\rangle \in R$ and $\langle y, z\rangle \in R$ always imply $\langle x, z\rangle \in R$.
R satisfies trichotomy on X iff for all $y, z \in X$, exactly one of the statements $\langle x, y\rangle \in R$, $x = y$, $\langle y, x\rangle \in R$ is valid.
R is an equivalence relation on X iff R is reflexive, symmetric and transitive on X.
R is a linear ordering on X iff R is transitive on X and satisfies trichotonomy on X.

Zorn's Lemma. Assume that $X \subseteq \mathbb{P}(Y)$. A subset Z of X is called a chain iff the subset-relation $\subset$ satisfies trichotonomy on Z. Now Zorn's Lemma says the following: If for every chain $Z \subseteq X$ it holds that $(\bigcup Z) \in X$, then there is an element $V \in X$ which is maximal, that is, there is no set $W \in X$ with $V \subset W$.

One says that X has at most as many elements as Y iff there is a one-one function $f : X \to Y$, that is, f maps the elements of X to elements of Y without repetition. Two sets X, Y have the same size iff X has at most as many elements as Y and Y has at most as many elements as X. The Schröder-Bernstein Theorem says that two sets have the same size iff there is a function $f : X \to Y$ which is one-one and onto.

Cardinal numbers. A cardinal is a set among all sets of the same size which represents the sets of this size; cardinals can therefore be viewed as both, numbers as sets. Usually, $\aleph_0$ represents the countable sets. For $n \in \mathbb{N}$, the number n represents the cardinality of the set $\{m \in \mathbb{N} : m < n\}$ or, more in general, n is the cardinal of all sets with n elements. When one assumes the Axiom of Choice, then also all uncountable

sets have a cardinal. One says that two cardinals κ and λ satisfy $\kappa \leq \lambda$ iff there is a one-one mapping from a set with cardinal κ to a set with cardinal λ. In general, one uses the following notions:

- $\mathrm{Card}(X)$ denotes the cardinal of X;
- $\mathrm{Card}(X) \leq \mathrm{Card}(Y)$ iff there is a one-one function with domain X and the range contained in Y;
- $\mathrm{Card}(X) = \mathrm{Card}(Y)$ iff there is a one-one function with domain X and range Y, that is, the function is a bijection from X onto Y;
- For those of you who know a bit of set theory and ordinals (which are generalisation of the natural numbers), if α is an ordinal then the cardinal α_α is the size of the smallest set X which satisfies $\mathrm{Card}(X) > \aleph_\beta$ for all $\beta < \alpha$ and $\mathrm{Card}(X) > n$ for all natural numbers n. Thus one has a hierarchy $0 < 1 < 2 < \ldots < \aleph_0 < \aleph_1 < \aleph_2 < \ldots \aleph_\omega < \aleph_{\omega+1} < \aleph_{\omega+2} < \ldots \aleph_{\omega \cdot 2} < \aleph_{\omega \cdot 2+1} < \aleph_{\omega \cdot 2+2}$ and this hierarchy goes beyond the size of any given set.

Continuum Hypothesis. One denotes the cardinal of the real numbers with $2^{\aleph_0}$ and one cannot determine its exact value. One only knows that whenever $2^{\aleph_0} = \aleph_{\omega \cdot n+m}$ then $m > 0$ and for all $n \in \mathbb{N}$ and $m \in \{1, 2, 3, \ldots\}$, this equation is consistent with the axioms ZFC of set theory. Cantor conjectured that $2^{\aleph_0} = \aleph_1$, but was not able to prove it and today one knows that one can neither prove nor disprove this conjecture. The conjecture is known as Continuum Hypothesis.

Schröder-Bernstein Theorem for Cardinals. If $\mathrm{Card}(X) \leq \mathrm{Card}(Y)$ and $\mathrm{Card}(Y) \leq \mathrm{Card}(X)$ then $\mathrm{Card}(X) = \mathrm{Card}(Y)$. In other words, if there are injective functions from X into Y and from Y into X then there is a bijective function between X and Y. Let κ, λ be cardinals. If $\kappa \leq \lambda$ and $\lambda \leq \kappa$ then $\kappa = \lambda$. The cardinals are linearly ordered (under the assumption of the axiom of choice).

Adding and Multiplying Cardinals. If X and Y are sets, then the sets $X \times \{0\}$ and $Y \times \{1\}$ are disjoint. Now one defines $\mathrm{Card}(X) + \mathrm{Card}(Y)$ as $\mathrm{Card}((X \times \{0\}) \cup (Y \times \{1\}))$. Furthermore, $\mathrm{Card}(X) \cdot \mathrm{Card}(Y)$ is $\mathrm{Card}(X \times Y)$. Note that these operations turn out to be independent on the representatives, thus they define arithmetic on

cardinals. The next result shows that the arithmetic on the infinite cardinals is quite easy.

Cardinal Arithmetic Theorem. If κ, λ are cardinal numbers and λ is infinite and $\kappa \neq 0$ then $\kappa + \lambda = \max\{\kappa, \lambda\}$ and $\kappa \cdot \lambda = \max\{\kappa, \lambda\}$. However, $0 \cdot \lambda = 0$ as for the natural numbers.

Theorem 0D. If X is an infinite set, then $\mathrm{Card}(X) = \mathrm{Card}(X^*)$, where X^* is the set of all finite sequences over elements of X.

Proof for $X = \mathbb{N}$. For each sequence $x = \langle x_1, x_2, \ldots, x_n \rangle$, one considers the set $\{x_1, x_1 + x_2 + 1, \ldots, x_1 + x_2 + \ldots + x_n + n - 1\}$ and one can see that there is a bijection between sequences of length n and n-element sets. The sequence of length 0 corresponds to $\emptyset$. So for each sequence, one identifies it with a finite set. Furthermore, one can map now a finite set D to the number $x = \sum_{y \in D} 2^y$; this is again a bijection. The concatenation of these two bijections gives a bijection from all finite sequences of natural numbers to the natural numbers. $\square$

An alternative coding, as used by Enderton, codes the sequence $\langle x_1, x_2, \ldots, x_n \rangle$ by the number $2^{x_1+1} \cdot 3^{x_2+1} \cdot 5^{x_3+1} \cdot \ldots \cdot p_n^{x_n+1}$. This coding, however, is only one-one but not a bijection: If in the range a number is divisible by 7 then this number is also divisible by 5.

Chapter 1.0

Example. One can formalise statements. For example, in case of chemical substances, one could list out for each chemical element whether it was detected in a substance. However, one might concentrate on important elements for the findings. Here some examples of summarised investigation results on substances:

1. $\mathbf{Na} \wedge \mathbf{Cl}$: The substance contains sodium (**Na**) and Chlorine (**Cl**);
2. $\mathbf{C} \wedge \mathbf{O} \wedge \mathbf{H}$: The substance contains carbon (**C**), oxygen (**O**) and hydrogen (**H**);
3. $\mathbf{Cl} \wedge \neg\mathbf{K}$: The substance contains chlorine but does not contain potassium;
4. $(\mathbf{Mg} \vee \mathbf{Ca}) \wedge \mathbf{F}$: The substance contained at least one of magnesium (**Mg**) and calcium (**Ca**); it furthermore also contained flourine;
5. $\mathbf{S} \rightarrow \mathbf{O}$: It was established that if the substance contained sulfur then it also contained oxygen;
6. $(\mathbf{S} \leftrightarrow \mathbf{H}) \wedge \mathbf{O}$: The substance contained oxygen (**O**); furthermore, it contained sulfur (**S**) if and only if it contained hydrogen (**H**).

The statements are given by atoms (which each say that the substance investigated contained a certain chemical element). These statements can be connected by connectives "and" ($\wedge$), "or" ($\vee$), "not" ($\neg$) and "if-then" ($\rightarrow$). In some cases, there are several ways to state the results formally. Both of the following sentences express the same fact:

1. $\neg(\mathbf{Cl} \vee \mathbf{K})$: It did not happen that the substance contained chlorine or contained potassium;
2. $\neg\mathbf{Cl} \wedge \neg\mathbf{K}$: The substance contained neither cholrine nor potassium.

Another example of this is the following statement which says that the substance contains at least two of the elements gold (**Au**), silver (**Ag**) and copper (**Cu**):

1. $(\mathbf{Au} \wedge \mathbf{Ag}) \vee (\mathbf{Au} \wedge \mathbf{Cu}) \vee (\mathbf{Ag} \wedge \mathbf{Cu})$;
2. $(\mathbf{Au} \vee \mathbf{Ag}) \wedge (\mathbf{Au} \vee \mathbf{Cu}) \wedge (\mathbf{Ag} \vee \mathbf{Cu})$.

One can use truth-tables to check whether two formulas are equivalent. For example, for the formulas $\neg(\mathbf{Cl} \vee \mathbf{K})$ and $\neg\mathbf{Cl} \wedge \neg\mathbf{K}$, one computes the values of these formulas for each combinations of values "false" and "true" for **Cl** and **K**. Then one will see that both formulas evaluate to "true" exactly when both atoms **Cl** and **K** are set to "false". Thus the formulas are equivalent.

Well-Formed Formulas. A well-formed formula (wff) uses the connectives $\rightarrow$, $\neg$ and brackets and atoms; while the atoms above where the abbrevations of chemical elements, one now uses just numbered items $A_1, A_2, \ldots$.

Well-formed formulas can be generated as follows: Every atom is a well-formed formula. Furthermore, when σ is a well-formed formula then one can any occurrence of an atom A_k in the formula replace by either $(\neg A_i)$ or $(A_i \rightarrow A_j)$ for some atoms A_i, A_j and the well-formed formulas are all those which can be constructed by finitely many of such replacements; note that not all occurrences of A_k need to be replaced, but only one and repeated replacements can use different strings to replace the text. So a well-formed formula is any sequence of brackets, atoms, $\neg$ and $\rightarrow$ which can be generated by finitely many of these replacements after starting with some atom.

Examples. The following are wffs: $(A_1 \rightarrow (A_5 \rightarrow (\neg A_6)))$; $((\neg A_2) \rightarrow (\neg A_7))$.

The following are not wffs, but can easily be made into some by introducing opening and closing brackets accordingly: $A_1 \rightarrow \neg A_2$; $(A_3 \rightarrow A_4) \rightarrow \neg A_5$. Informally, such formulas are used whenever it is clear for what wff they stand. Here the first of these examples would stand for $(A_1 \rightarrow (\neg A_2))$ and the second for $((A_3 \rightarrow A_4) \rightarrow (\neg A_5))$.

The following are definitely not wffs and such formulas should also not be used. $)A_1 \rightarrow A_2(; \neg(A_1 \neg \rightarrow A_2)$. The first of these has only the opening and closing bracked mixed up, for the second it is even more difficult to see. Similarly formulas without balanced brackets are also invalid.

Formal languages in computing. Formal languages show up in many contexts, not only in logic. In particular programming languages employ a lot of formal languages and computer science has come up with good algorithms and computer programs to check whether a given text (of a program) adhers to the rules of a given programming language and when not, the user is provided with information what could be wrong.

Chapter 1.1 – The Language of Sentential Logic

The letters of the language and their meaning. The logical language uses atoms $A_1, A_2, A_3, \ldots$ and connectives $\rightarrow, \leftrightarrow, \wedge, \vee, \neg$ and brackets and truth-constants 0 (false) and 1 (true). Each of the connectives has a meaning which can be summarised by the following truth-table:

sent. symbol	sent. symbol	implication	equivalence	conjunction	disjunction	negated equivalence	negation
atom	atom	implies	equals	and	or	exclusive or	not
A_1	A_2	$A_1 \rightarrow A_2$	$A_1 \leftrightarrow A_2$	$A_1 \wedge A_2$	$A_1 \vee A_2$	$A_1 \oplus A_2$	$\neg A_1$
0	0	1	1	0	0	0	1
0	1	1	0	0	1	1	1
1	0	0	0	0	1	1	0
1	1	1	1	1	1	0	0

The letters of the language are also called symbols. Each symbol has the length 1, also every atom and every bracket. So the formula $(\neg A_3)$ has the length 4 and the formula $(A_3 \vee A_8)$ has the length 5. Finite sequences of symbols are called expressions and those which are meaningful are called formulas. If they in addition also adher to the syntactic requirements strictly, they are called well-formed formulas.

Formula-Constructor-Functions. One can define functions which put formulas together to get a new one. For example, $\mathcal{E}_{\rightarrow}(\alpha, \beta) = (\alpha \rightarrow \beta)$. Similarly one defines $\mathcal{E}_{\neg}(\alpha)$ and $\mathcal{E}_{\leftrightarrow}(\alpha, \beta)$ and so on. Then a wff is just any formula which can be assembled out of atoms and truth-constants by finitely often using these functions. A construction sequence of a formula is a sequence of expressions such that every experssion is either an atom or obtained from prior expressions by one of the above mentioned functions. So for $((A_1 \leftrightarrow A_2) \vee (\neg A_5))$, the corresponding construction sequence can be A_1, A_2, $(A_1 \leftrightarrow A_2)$, A_5, $(\neg A_5)$, $((A_1 \leftrightarrow A_2) \vee (\neg A_5))$. Note that this sequence is not unique, as sometimes the order of some parts can be exchanged, so one could start with A_1, A_2 or with A_2, A_1 and then let the rest of the sequence follow.

Induction Principle. If a set of formulas S contains all atoms and truth-constants and is closed under $\mathcal{E}_{\neg}$, $\mathcal{E}_{\rightarrow}$, $\mathcal{E}_{\leftrightarrow}$, $\mathcal{E}_{\wedge}$, $\mathcal{E}_{\vee}$, $\mathcal{E}_{\oplus}$ then S contains every wff.

Proof. Assume that a formula α together with its construction sequence $\beta_1, \beta_2, \ldots, \beta_n$ is given. Every β_m is either an atom or obtained by some construction function from β_i, β_j with $i, j < m$. It follows then from the closure of S under all construction functions and from the fact that S contains all atoms, that whenever all formulas β_k with $k < m$ are wffs, so is β_m. Thus one obtains from ordinary induction that all β_m are in S and hence $\alpha = \beta_n$ is in S. $\square$

Example. Let S be the set of all expressions which have the same amount of opening and closing brackets. Then S contains all wffs.

Proof. All atoms are in S, as the amount of opening and the amount of closing bracktes in atoms is both zero. Furthermore, if $\alpha, \beta \in S$ then there are i and j such that α has i opening and i closing brackets and β has j opening and j closing brackets. Now the following formulas have $i + j + 1$ opening and $i + j + 1$ closing brackets: $(\alpha \to \beta)$, $(\alpha \leftrightarrow \beta)$, $(\alpha \wedge \beta)$, $(\alpha \vee \beta)$, $(\alpha \oplus \beta)$; furthermore, the formula $(\neg\alpha)$ has $i + 1$ opening and $i + 1$ closing brackets. Thus the constructor functions map formulas in S to formulas in S and S is closed under all constructor functions. It follows that S contains all wffs. $\square$

Chapter 1.2 – Truth Assignments

Logic and Values. Within this course, the topic is $\{0,1\}$-valued logic. Other people also investigate $\{0,1,u\}$-valued logic where "u" stands for unknown. Then whenever the value of an operation cannot be determined, it is u: For example, $0 \wedge u = 0$ as the and ($\wedge$) can only be true when both operands are 1 or might be 1; however, $u \wedge u = u$ as one has no idea what the two inputs u of the and stand for. There were even logics with more than three values investigated and also the meaning of the third value can be different between different versions of three-valued logics. From now on, the only truth-values are 0 and 1.

Main Idea. A truth-assignment ν is a mapping which assigns to every atom a truth-value (0 or 1). The goal is to extend the mapping ν to all well-formed formulas.

Example. If $\nu(A_1) = 1$, $\nu(A_2) = 0$ and $\nu(A_3) = 1$ then one the goal would be to construct an extension $\overline{\nu}$ of ν to all wff which would be consistent with the definitions of connectives, for example, $\overline{\nu}((A_1 \wedge A_2) \vee (\neg A_3)) = 0$.

Inductive Definition of Truth-Assignment. If ν is a truth-assignment on all atoms, one defines inductively $\overline{\nu}$ on all wffs:

1. $\overline{\nu}(0) = 0$, $\overline{\nu}(1) = 1$ and $\overline{\nu}(A_k) = \nu(A_k)$;
2. $\overline{\nu}(\neg\alpha) = 1 - \overline{\nu}(\alpha)$;
3. $\overline{\nu}(\alpha \wedge \beta) = \overline{\nu}(\alpha) \cdot \overline{\nu}(\beta)$;
4. $\overline{\nu}(\alpha \vee \beta) = \overline{\nu}(\alpha) + \overline{\nu}(\beta) - \overline{\nu}(\alpha) \cdot \overline{\nu}(\beta)$;
5. $\overline{\nu}(\alpha \to \beta) = \overline{\nu}((\neg\alpha) \vee \beta)$;
6. $\overline{\nu}(\alpha \oplus \beta) = \overline{\nu}(\alpha) + \overline{\nu}(\beta) - 2 \cdot \overline{\nu}(\alpha) \cdot \overline{\nu}(\beta)$;
7. $\overline{\nu}(\alpha \leftrightarrow \beta) = \overline{\nu}(\neg(\alpha \oplus \beta))$.

These arithmetic formulas just say that $\overline{\nu}$ of a connective of two formulas α and β combines $\overline{\nu}(\alpha)$ and $\overline{\nu}(\beta)$ exactly in the way as the truth-table for the connective prescribes.

Example. Consider the formula $(((A_1 \to A_2) \vee (A_3 \leftrightarrow A_4)) \wedge (\neg A_5))$ and the following two truth-assignments μ and ν:

- $\overline{\mu}(A_1) = 1$, $\overline{\mu}(A_2) = 1$, $\overline{\mu}(A_3) = 1$, $\overline{\mu}(A_4) = 0$ and $\overline{\mu}(A_5) = 0$.
- This implies $\overline{\mu}(A_1 \to A_2) = 1$, $\overline{\mu}(A_3 \leftrightarrow A_4) = 0$, $\overline{\mu}((A_1 \to A_2) \vee (A_3 \leftrightarrow A_4)) = 1$, $\overline{\mu}(\neg A_5) = 1$ and $\overline{\mu}(((A_1 \to A_2) \vee (A_3 \leftrightarrow A_4)) \wedge (\neg A_5)) = 1$.
- $\overline{\nu}(A_1) = 1$, $\overline{\nu}(A_2) = 0$, $\overline{\nu}(A_3) = 1$, $\overline{\nu}(A_4) = 0$ and $\overline{\nu}(A_5) = 1$.
- This implies $\overline{\nu}(A_1 \to A_2) = 0$, $\overline{\nu}(A_3 \leftrightarrow A_4) = 0$, $\overline{\nu}((A_1 \to A_2) \vee (A_3 \leftrightarrow A_4)) = 0$, $\overline{\nu}(\neg A_5) = 0$ and $\overline{\nu}(((A_1 \to A_2) \vee (A_3 \leftrightarrow A_4)) \wedge (\neg A_5)) = 0$.

In short words: After knowing μ on all relevant atoms, one can determine $\overline{\mu}$ on all subformulas in the constructor sequence inductively in order to get $\overline{\mu}$ for the full formula. Similarly for ν and $\overline{\nu}$. One can also write this in short-hand.

$$
\begin{array}{lcccccccccc}
 & ((& (A_1 & \to & A_2) & \vee & (A_3 & \leftrightarrow & A_4)) & \wedge & (\neg A_5)) \\
\overline{\mu}\colon & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\
\overline{\nu}\colon & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1
\end{array}
$$

So the values of the truth-assignment stand below the atom or below the connective; in the latter case they are the value which the connective takes during evaluating the formula.

Evaluation of truth-assignments by string-operations. Another method to evaluate the truth-assignment is to take a formula α and do any of the below as long as it applies:

1. If there is an atom A_k in the formula, replace A_k by the value $\nu(A_k)$;
2. If there is a subformula of one of the following types, replace it by 1: $(\neg 0)$, $(1 \vee 0)$, $(0 \vee 1)$, $(1 \vee 1)$, $(1 \wedge 1)$, $(0 \leftrightarrow 0)$, $(1 \leftrightarrow 1)$, $(0 \to 0)$, $(0 \to 1)$, $(1 \to 1)$, $(0 \oplus 1)$, $(1 \oplus 0)$;
3. If there is a subformula of one of the following types, replace it by 0: $(\neg 1)$, $(0 \vee 0)$, $(0 \wedge 0)$, $(0 \wedge 1)$, $(1 \wedge 0)$, $(0 \leftrightarrow 1)$, $(1 \leftrightarrow 0)$, $(1 \to 0)$, $(0 \oplus 0)$, $(1 \oplus 1)$.

Once none of the rules applies, the formula is either 0 or 1 and that value is $\overline{\nu}(\alpha)$.

Example. Consider the formula $((A_1 \to A_2) \vee (A_3 \leftrightarrow A_4))$ and the following truth-assignment μ:

- $\mu(A_1) = 1$, $\mu(A_2) = 1$, $\mu(A_3) = 1$ and $\mu(A_4) = 0$.
- Now the algorithm does the following replacements:
  $((A_1 \to A_2) \vee (A_3 \leftrightarrow A_4))$;
  $((1 \to A_2) \vee (A_3 \leftrightarrow A_4))$;
  $((1 \to 1) \vee (A_3 \leftrightarrow A_4))$;
  $(1 \vee (A_3 \leftrightarrow A_4))$;
  $(1 \vee (1 \leftrightarrow A_4))$;
  $(1 \vee (1 \leftrightarrow 0))$;
  $(1 \vee 0)$;
  1.

Note that the order of the replacements is not unique; for example one could have replaced all atoms first and then done the connectives.

Theorem 12A. For every truth assignment ν there is a unique function $\overline{\nu}$ which assigns to all wff a unique truth-value respecting all the connectives and the values of ν at the atoms.

The proof will follow in Chapters 1.3 and 1.4. However, above examples already indicate that it is true. One uses the following definition for further studies.

Definition. If X is a set of wffs and α a formula, one says that X tautologically implies α (written $X \models \alpha$) iff every truth assignment ν for atoms occurring in X or α satisfies that whenever $\overline{\nu}(\beta) = 1$ for all $\beta \in X$ then $\overline{\nu}(\alpha) = 1$. One says that α is a tautology iff $\emptyset \models \alpha$.

Examples. The following are examples for $\models$.

1. $\{A_1, (\neg A_1)\} \models A_2$: There is a general principle behind it: if a set X is not made true by any truth-assignment, it tautologically implies every formula, even false ones.
2. $\{A_1, (A_1 \to A_2)\} \models A_2$: If A_1 is true and also $A_1 \to A_2$ is true, then A_2 is also true.

3. $\{A_1, A_2\} \models (A_1 \wedge A_2)$: If A_1 and A_2 are both true, then is also their conjunction.

4. $\{A_1\} \models (A_1 \vee A_2)$: If A_1 is true, then also the disjunction of A_1 with whatsoever.

One says that a set of formulas X is satisfiable iff there is a truth-assignment ν such that $\overline{\nu}(\alpha) = 1$ for all $\alpha \in X$. One of the goals of Chapter 1 is also to prove the following compactness theorem.

Compactness Theorem. If X is an infinite set of formulas such that every finite subset $Y \subseteq X$ is satisfiable then X itself is also satisfiable.

The Truth-Table Method. In order to compare two formulas α and β in atoms $A_1, \ldots, A_k$, one marks down the 2^k combinations of truth-values and then compares the values for α and β. Examples are given which are optimised as follows:

1. In an $\wedge$ or $\vee$, if one side of the connective determines the value, the other side can be ignored and therefore not all different possibilities of the atoms need to be listed;

2. In some cases, the formulas employ an atom as a truth-variable in order to cut down on the number of cases.

The following formulas are investigated.

1. De Morgan Law: $(\neg(A_1 \wedge A_2)) = ((\neg A_1) \vee (\neg A_2))$.

2. Distributivity: $(A_1 \vee (A_2 \wedge A_3)) = ((A_1 \vee A_2) \wedge (A_1 \vee A_3))$.

3. Chain Implication: $((((A_1 \wedge A_2) \to A_3) \to A_4) \to ((A_1 \to A_3) \to A_4))$ is a tautology.

4. Stronger Antedecent: $\{(A_1 \wedge A_2)\} \models A_1$; $\{(A_1 \to A_3)\} \models ((A_1 \wedge A_2) \to A_3)$.

The proofs of the first two items are standard by truth-tables. The fourth item is easy to see. The first formula of the fourth item is clear, the second formula of the fourth item satisfies the following: the right side can only be false when $(A_1 \wedge A_2)$ is true and A_3 is false; however, then also A_1 is true and therefore $(A_1 \to A_3)$ is false as well, so that the second statement is also right in this case.

For the third item, let $\alpha = (\gamma \to \delta)$ and $\beta = (\delta \to A_4) \to (\gamma \to A_4)$, where γ and

δ are determined later. Now assume that α is true. If A_4 is true, then $(\delta \to A_4)$ and $(\gamma \to A_4)$ are both true and β is true. If A_4 is false and γ is true, then also δ is true by α and therefore β is the implication $(1 \to 0) \to (1 \to 0)$ which is true. If γ is false, then there is $(0 \to 0)$ on the right side of β and again β is true. Thus $\{\alpha\} \models \beta$.

Now let $\gamma = (A_1 \to A_2) \to A_3$ and $\delta = (A_1 \to A_3)$. By the fourth item, it holds that $\{\gamma\}$ tautologically implies δ. This is equivalent to saying that $(\gamma \to \delta)$ is a tautology. Thus α is a tautology. By the previous argument, it follows that β is also a tautology. As β is the statement of the third item, the third item is proven.

Selected list of tautologies.

1. Associative and commutative laws for $\wedge$, $\vee$, $\leftrightarrow$, $\oplus$; for example $((A_1 \wedge A_2) \leftrightarrow (A_2 \wedge A_1))$ and $(((A_1 \wedge A_2) \wedge A_3) \leftrightarrow (A_1 \wedge (A_2 \wedge A_3)))$. These rules are also done with the other connectives and one can also do the following: For every k and every wff α, one can simultaneously replace all occurrences of A_k in the preceding formulas by α.

2. Distributive laws for $\vee$ versus $\wedge$, $\wedge$ versus $\vee$ and $\wedge$ versus $\oplus$. Here examples for $\wedge$ versus $\oplus$: $((A_1 \wedge (A_2 \oplus A_3)) \leftrightarrow ((A_1 \wedge A_2) \oplus (A_1 \wedge A_3)))$ and $(((A_1 \oplus A_2) \wedge A_3) \leftrightarrow ((A_1 \wedge A_3) \oplus (A_2 \wedge A_3)))$.

3. Double negation: $(A_1 \leftrightarrow (\neg(\neg A_1)))$;
   Negation formula for $\to$: $((\neg(A_1 \to A_2)) \Leftrightarrow (A_1 \wedge (\neg A_2)))$;
   Negation formulas for $\leftrightarrow$: $((\neg(A_1 \leftrightarrow A_2)) \leftrightarrow (A_1 \oplus A_2))$, $((\neg(A_1 \leftrightarrow A_2)) \leftrightarrow ((\neg A_1) \leftrightarrow A_2))$ and $((\neg(A_1 \leftrightarrow A_2)) \leftrightarrow (A_1 \leftrightarrow (\neg A_2)))$;
   De Morgan's Laws: $((\neg(A_1 \wedge A_2)) \leftrightarrow ((\neg A_1) \vee (\neg A_2)))$ and $((\neg(A_1 \vee A_2)) \leftrightarrow ((\neg A_1) \wedge (\neg A_2)))$.

4. Excludced middle: $(A_1 \vee (\neg A_1))$.

5. Contradiction: $(\neg(A_1 \wedge (\neg A_1)))$.

6. Contraposition: $((A_1 \to A_2) \leftrightarrow ((\neg A_2) \to (\neg A_1)))$.

7. Exportation: $(((A_1 \wedge A_2) \to A_3) \leftrightarrow (A_1 \to (A_2 \to A_3)))$.

Chapter 1.3 – A Parsing Algorithm

Note that it was proven in the last example of Chapter 1.1 that a wff has the same amount of opening and closing brackets. This is now stated as a lemma.

Lemma 13A. Every wff has the same number of opening and closing brackets.

The following lemma deals with the order of brackets in wffs.

Lemma 13B. If a wff is split into two non-empty expressions α and β then α has more opening than closing brackets and β has more closing than opening brackets.

Proof. One shows this by structural induction. Atoms and truth-values are one symbol and cannot be split into two non-empty parts; thus in the base case the statement is vacuously true. Assume now that it is proven for γ and δ and that one looks at the formulas $(\neg\gamma)$ and $(\gamma \to \delta)$, where, if needed, a subformula γ or δ of these formulas is split into non-empty parts α', β'. One obtains the following possible splittings of the formulas:

$(\neg\gamma)$ The possible splittings are (and $\neg\gamma$), ($\neg$ and γ), ($\neg\alpha'$ and β'), ($\neg\gamma$ and).
In all cases, the left side contains the outmost opening bracket of the formula plus some symbols plus either γ where the brackets are balanced or α' which contains more opening than closing brackets. Thus the left side contains more opening than closing brackets and similarly one shows that the right side contains more closing than opening brackets.

$(\gamma \to \delta)$ The possible splittings are (and $\gamma \to \delta$), (α' and $\beta' \to \delta$), (γ and $\to \delta$), ($\gamma \to$ and δ), ($\gamma \to \alpha'$ and β'), ($\gamma \to \delta$ and).
As above, one can verify that in all cases the left side has more opening than closing brackets and the right side has more closing than opening brackets. This uses that the same is true for the parts α' and β' considered, that γ and δ have balanced brackets and that the outmost opening bracket is on the left side and the outmost closing bracket is on the right side without being balanced off inside their sides.

These arguments prove the lemma. □

A Parsing Algorithm. There is an implementation of a parting algorithm for this course. It is on the page

http://www.comp.nus.edu.sg/~fstephan/parsing.html

and the source code of the parser is the following one:

```
function display() // Prepares a string containing the values of variables
  { var t="";
    for (index in atoms)
      { t = t+index+atoms[index]; }
    return(t); }

function ad(t,a) // outputs messages with "alert" and "document.write"
  { if (a>1) { alert(t); }
    if (a>1) { document.write(t); }
    else { document.write(formula); }
    document.write("<br>"+display()+"<br>"); }

function subparse() // subroutine to parse the formula
  { var a = 0; var b = 0;
    if (formula[formulapos] == "(")
      { formulapos++;
        if (formula[formulapos] == "!") // subcase of negation
          { formulapos++; a = subparse();
            if (a > 1) { return(a); }
            if (formula[formulapos] != ")")
              { return(2); }
            formulapos++; return(1-a); }
        a = subparse(); //subcase of (a operator b)
        var c = formula[formulapos];
        if ((c!="&")&&(c!="|")&&(c!="+")&&(c!="<")&&(c!=">")&&(c!="="))
          { return(3); }
        formulapos++;
        b = subparse();
        if (b>1) { return(b); }
        if (formula[formulapos]!=")")
          { return(2); }
        formulapos++;
        switch(c)
          { case "&": return(a&b);
            case "|": return(a|b);
```

```
            case "+": return(a^b);
            case ">": return(1-a+a*b);
            case "<": return(1-b+a*b);
            case "=": return(1-(a^b));
            default: return(3); } }
    if (formula[formulapos]=="0") // subcase of constant 0
      { formulapos++; return(0); }
    if (formula[formulapos]=="1") // subcase of constant 1
      { formulapos++; return(1); }
    if (formula[formulapos] in atoms) //subcase of atom-value
      { index = formula[formulapos]; formulapos++;
        return(atoms[index]); }
    return(5); }

function parse()
  { var index = formula[0]; var a=4;
    if (formula.length == 0) { a = 6; }
    else if ((index in atoms)&&(formula[1]=="="))
      { formulapos = 2;
        var a = subparse(); }
    switch(a) // generating text for error-messages
      { case 0: atoms[index] = 0; ad("",a); break;
        case 1: atoms[index] = 1; ad("",a); break;
        case 2: ad("Bracket Error at position "+formulapos+
                " of "+formula,a); break;
        case 3: ad("Unknown Operator at position "+
                formulapos+" of "+formula,a); break;
        case 4: ad("Formula value not assigned to a variable",a); break;
        case 5: ad("Variable or constant does not exist at"+
                  " position "+formulapos+" of "+formula,a); break; }
    if ((a<2)&&(formula.length!=formulapos))
      { alert("Symbols after end of formula ignored"); }
    return(a); }

// main part of program: initialisation followed by main loop
var atoms = new Array();
var letters = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
var index; var indexpos; var formula; var formulapos = 0;
```

```
for (indexpos=0;indexpos<letters.length;indexpos++)
  { index = letters.charAt(indexpos); atoms[index] = 0; }
do
  { formula = prompt(display()+" Input Formula"); } // reading formula
while (parse()!=6); // do the parsing and end when formula is empty
```

Note that this parser uses for all connectives special symbols which are explained on the page of the parser. Besides parsing the formula, it also assigns a value, while taking the stored values of the atoms into account; these values are initially 0 and can be modified by previously processed formulas. There are 26 atoms, each letter is one of them.

The main part of the parser is in the function "subparse()" in which it enters and where the parameters are in the global variables "formula" and "formulapos" (current position in formula) and "atoms" (current values of the atoms) and "letters" (possible names of atoms) and "index" (variable to access atoms) and "indexpos" (position of index in the list of all possible indices). In the function subparse, the parser distinguishes the following cases:

1. The current subformula is a truth-value or atom: Then it returns its value;
2. The current subformula starts with "(" followed by a negation symbol: Then the parser reads the negation symbol, calls itself for processing the subformula and recieving its value and then passes on the negated value of it while verifying that the closing bracket is there and advancing the position beyond it;
3. The current subformula starts with "(" followed by something what is not a negation symbols: Then the formula is of the form $(\alpha + \beta)$ – or with another operator instead of the implication – and after processing the opening bracket, the subparse calls itself to process α and obtain its value and it then reads out the symbol representing the binary operation and then it again calls itself to process β and obtain its value and then the parser forms the exclusive or of the values of α and β and returns this value.
4. Whenever there is an error occurring during the processing, the parser aborts and returns an error code which it at least 2 (to distinguish it from truth-values).

The main program assigns the value obtained by subparse() to the current atom of the line and then goes on with processing the next user command. All commands are assignments like

```
A=((B+C)+1)
```

where, as said, the plus stands for exclusive or. Also the other Boolean operations are expressed by single symbols used in common programming languages.

An inspection of the program shows that the parser and its subfunction evaluate all the subformulas by the conditions on their evaluation in the "Inductive Definition of the Truth-Assignments" (though it uses Boolean Java Script operations rather than putting everything into arithmetics). Thus, if one has set the atoms with the values of ν, then the parser indeed returns $\overline{\nu}(\alpha)$ for a formula α typed into the input.

Instead of evaluating the formula, Enderton's parser produced a tree representing the formula; this tree can then be evaluated from the leaves to the root in order to get the formula. He in particular notes the following:

- Every constructor sequence of a wff α has the property that two subformulas β, γ on the tree with the property that whenever β is a subformula of γ, that is, that γ is on the branch leading from the root to β, then β comes in the constructor sequence before γ;

- If one evaluates the formulas on the tree from the leaves towards the root, any order is fine as long as no subformula is evaluated after its motherformula and thus, all orders which correspond to constructor sequences produce the same results.

A formal proof of the independence of truth-values on the chosen constructor-sequence will be given in Chapter 1.4.

Polish Notation. Polish notation avoids brackets by putting the connectives first and then let the operands follow. So one has again constructor functions which compute the following constructions:

1. $\alpha \mapsto \neg\alpha$;
2. $\alpha, \beta \mapsto \wedge\alpha\beta$;
3. $\alpha, \beta \mapsto \vee\alpha\beta$;
4. $\alpha, \beta \mapsto \oplus\alpha\beta$;
5. $\alpha, \beta \mapsto \rightarrow \alpha\beta$;

6. $\alpha, \beta \mapsto \leftrightarrow \alpha\beta$;

One can then define the formula along a construction sequence in Polish notation instead of the usual notation to get the same formula translated. Here an example:

Formula	Text	Normal Notation	Polish Notation
α_1	Atom	A_1	A_1
α_2	Atom	A_2	A_2
α_3	Atom	A_3	A_3
α_4	α_1 and α_2	$(A_1 \wedge A_2)$	$\wedge A_1 A_2$
α_5	Not α_3	$(\neg A_3)$	$\neg A_3$
α_6	α_4 or α_5	$((A_1 \wedge A_2) \vee (\neg A_3))$	$\vee \wedge A_1 A_2 \neg A_3$
α_7	α_1 eor α_3	$(A_1 \oplus A_3)$	$\oplus A_1 A_3$
α_8	α_6 implies α_7	$(((A_1 \wedge A_2) \vee (\neg A_3)) \vee (A_1 \oplus A_3))$	$\rightarrow \vee \wedge A_1 A_2 \neg A_3 \oplus A_1 A_3$

Formulas in Polish notation are more difficult to read for humans, partly also because humans are not trained to do so, unless they write logic textbooks. However, parsers can be written more easily (with less effort) for formulas in Polish notation. Furthermore, if one makes proofs over formulas in Polish notation, there are less case distinctions on the level of a parser. For that reason, Polish notation gained some popularity in mathematics and computer science.

Omitting Brackets. For human processing (and for sufficiently advanced computer programs and compilers), one can try to find an informal notation which allows to reduce the number of brackets of normal notation without having to use Polish notation. The idea is that there are connectives which bind more and connectives which bind less; thus when one puts together different connectives like in

$$A_1 \wedge A_2 \vee \neg A_3$$

then $\neg$ and $\wedge$ have priority over $\vee$ and thus one can put brackets in a way which respect this priority. The resulting formula is

$$((A_1 \wedge A_2) \vee (\neg A_3))$$

and can be evaluated by the parser. The rules are the following ones:

1. The outmost brackets can be omitted;

2. The negation symbol binds to what follows directly it, that is either an atom or a truth-value or an expression in brackets and double negations are to be removed from the formula as they do not have any effect (so that never a negation follows immediately a negation);

3. conjunction binds more than disjunction and both bind more than implication and these three all bind more than equivalence;

4. Two connectives of the same type between subformulas like, for example, $\alpha \to \beta \to \gamma$, where everything inside these formulas binds more than the connectives are bracketed as $\alpha \to (\beta \to \gamma)$.

Note that the fourth rule is only relevant for $\to$, as the other connectives $\wedge$, $\vee$, *oplus* and $\leftrightarrow$ are transitive and therefore there is no need to fix the rules for bracketing expressions only having one type of connectives. It is important to note that on one hand $\leftrightarrow$ is associative, but on the other hand the statement

$$\alpha \leftrightarrow \beta \leftrightarrow \gamma$$

is not equivalent to "either α, β, γ are all three true or α, β, γ are all three false". This contradicts to the everyday usage in mathematical writing.

Chapter 1.4 – Induction and Recursion

Induction is a proof-method which goes proves an assertion first on the base-cases and then by inductive steps on objects (numbers, formulas, strings) derived from the bases-cases in a finite number of steps. Recursion deals with a similar process where one defines a function first on the base-cases and then extends this definition to objects which are built from the base-cases in finitely many steps using constructor-operators. For this one needs to prove that this building-up of the function is unique: Either one proves that a certain uniqueness condition on the recursion is satisfied or one proves that different ways of extending the the base-cases to the target object through different intermediate results give the same value. Both approaches are valid; however, the second one is more work, as one has to do this prove each time one makes a function by recursion while in the first approach, it is sufficient to once prove that there is only a unique way to build the objects from the base-cases.

Example: Natural Numbers. Both recursion and induction originate from the study of natural numbers. These are given by the following: A set of numbers $\mathbb{N}$ together with a constant 0 and a constant 1 and addition $+$ and an order $<$ defined as

$$x < y \Leftrightarrow \exists z\,[x + z + 1 = y].$$

Now one can define multiplication by recursion as follows:

Base Case: $x \cdot 0 = 0$;

Recursion: $x \cdot (y + 1) = (x \cdot y) + 1$.

Furthermore, one can make an inductive prove that, for example, every number is even or odd. Here x is even iff x is of the form $y + y$ and x is odd iff x is of the form $y + y + 1$ for some y.

Base Case: $0 = 0 + 0$; hence 0 is even.

Inductive Step: There are two subcases, namely the one where x is even and the one where x is odd.

Even x: Then there is a y such that $x = y + y$. Now $x + 1$ is $y + y + 1$ and $x + 1$ is odd.

Odd x: Then there is y such that $x = y + y + 1$. Then $x + 1 = y + y + 1 + 1 = (y + 1) + (y + 1)$. So $x + 1$ is even.

Conclusion: When x is even then $x + 1$ is odd and when x is odd then $x + 1$ is even; hence when x is even or odd, so is $x + 1$. This completes the inductive step.

In short, one first proves the assertion for the base case $x = 0$ and then, for every x where the assertion is true, one also proves it for $x + 1$. The natural numbers can, more abstractly, viewed to be built up as follows:

A base set: This base-set is $\{0\}$;

A constructor function: This function is $x \mapsto x + 1$.

One proves an assertion now by being true on the base-set and then one shows that whenever it is true for some x, then it is also true for the image of x under any constructure function – here is only one of them, namely $x \mapsto x + 1$, but for formulas, there are several of them. Furthermore, one defines multiplication $x, y \mapsto x \cdot y$ first for the base-case where $y = 0$ and then by recursion for each expression $x \cdot y$, one extends the definition to $x \cdot (y + 1)$ by mapping it back to already known constructs like $x \cdot y$ and expressions using $+$.

Base-Cases, Constructor Functions and Sequences, Lower and Upper Limits. In the following let B be a base-set (like the set $\{0\}$ in the case of natural numbers) and $f : D \times D \to D$ and $g : D \to D$ be constructor functions for some set $D \supseteq B$; note that one can have any number of such functions and for the case of the natural numbers it is just one ($x \mapsto x + 1$). Now one defines the following notions:

Inductive set: A set $S \subseteq D$ is called *inductive* iff $B \subseteq S$ and for all constructor functions, they map members of S to members of S.

Upper Limit C^*: C^* is the intersection of all inductive sets S; as D is an inductive set, this intersection is not empty and so the result C^* is a subset of D which contains B. Furthermore, for all $x, y \in C^*$ and all inductive sets S, $x, y \in S$ and $f(x, y) \in S$ and $g(x) \in S$; hence one can conclude (as the choice of S was an arbitrary inductive set) that $f(x, y) \in C^*$ and $g(x) \in C^*$. Furthermore, as x, y was chosen arbitrarily, C^* is closed under f and g and also an inductive set. It is the smallest inductive set.

Constructor Sequences of length n for some $y \in D$: A sequence $x_1, x_2, \ldots, x_n$ is called *a construction sequence of length n* iff all x_m satisfy that either x_m is a member of B or there are members x_i, x_j of the sequence with $i, j < m$ such that either $x_m = f(x_i, x_j)$ or $x_m = g(x_i)$; such a sequence is called *a construction sequence of length n for y* iff it in addition satisfies that $y \in \{x_1, x_2, \ldots, x_n\}$.

Lower Limit C_*: The lower limit C_n is the set of all $y \in D$ such that there is an n and a construction sequence of length n for y; equivalently one can denote with C_n all the y for which there is a construction-sequence of length n and then let $C_* = \bigcup_{n \in \mathbb{N}} C_n$; note that here $C_1 = B$.

Note that in this definitions, C_* is an inductive set: The reason is that whenever $x, y \in C_*$ then there are construction sequences for x and for y. Now one can concatenate the construction sequences to and append at this sequence either $f(x, y)$ or $g(x)$ in order to prove that $f(x, y), g(x)$ are also in C_*. So C_* is an inductive set and therefore $C^* \subseteq C_*$.

Furthermore, one can make a proof that every member of C_* is contained in every inductive set S. So assume that $y \in C_*$ as witnessed by the construction sequence $x_1, x_2, \ldots, x_n$. Now one proves the following statement for all $m \leq n$:

> All x_k with $k \leq m$ are in S.

This statement is true for $m = 0$, as there is no x_0. Now assume that the statement is true for m and one wants to prove it for $m + 1$. For this, one has only to show that under the assumption that all x_k with $k \leq m$ are in S, then also $x_{m+1} \in S$. There are several cases:

1. If $x_{m+1} \in B$ then $x_{m+1} \in S$ by the fact that every inductive set contains the base set B;
2. If $x_{m+1} = f(x_i, x_j)$ for $i, j \leq m$ then $x_i, x_j \in S$ by induction hypothesis and the $x_{m+1} \in S$ follows by the fact that S is closed under f;
3. If $x_{m+1} = g(x_k)$ for a $k \leq m$ then $x_k \in S$ by induction hypothesis and $x_{m+1} \in S$ follows by the fact that S is closed under g.

One of these cases does always apply by the definition of a construction sequence and therefore all members of the construction sequence are in S. Thus $C_* \subseteq S$ for every inductive set S and so also $C_* \subseteq C^*$.

Induction Principle. $C_* = C^*$ by the above proof and in the following, C denotes either of them. Furthermore, if S is an inductive subset of C (that is, a set which contains B and is closed under all constructor functions) then $S = C$.

Examples. Here are several examples of sets defined as the closure of a base set B under certain operations; note that each of them needs a universe D on which the operations are defined. For the below, the universe is always $D = \mathbb{Q}$ the set of rational numbers.

1. Consider the base set $B = \{0\}$ and the function $g(x) = x + 1$. Then the closure of B under g is the set $C = \mathbb{N}$ of all natural numbers.

2. Consider the base set $B = \{0, 1\}$ and the constructor functions $f(x, y) = x + y$ and $g(x) = -x$. Then the closure of B under f and g is the set $C = \mathbb{Z}$ of all integers.

3. Consider the base set $B = \mathbb{Z}$ and one constructor function f given as $f(x, y) = (x + y)/2$. Then the set C is the set of all dyadic rationals, that is, all rational numbers which can be represented such that the denominator is a power of 2.

4. Consider the basis set $B = \{0, 1\}$ and the constructor functions $x, y \mapsto x + y$, $x \mapsto -x$ and $x, y, z, u \mapsto 1/(1 + x^2 + y^2 + z^2 + u^2)$. The resulting closure of B under these functions is $C = \mathbb{Q}$.

Furthermore, if one fixes a set B consisting of all atoms $A_1, A_2, \ldots$ and the truth-values 0 and 1 and if one lets D to be the set of all strings (expressions) over an alphabet consisting of the symbols in B and the brackets and the logical connectives and if one takes F to be the set of constructor functions $\alpha \mapsto (\neg\alpha)$, $\alpha, \beta \mapsto (\alpha \wedge \beta)$, $\alpha, \beta \mapsto (\alpha \vee \beta)$, $\alpha, \beta \mapsto (\alpha \oplus \beta)$, $\alpha, \beta \mapsto (\alpha \to \beta)$ and $\alpha, \beta \mapsto (\alpha \leftrightarrow \beta)$, then C consists of all wffs including those which contain truth-values as constants somewhere in the formula.

Definition: Subformulas. Given a wff $\alpha = a_1 a_2 \ldots a_n$, β is a subformula of α iff β is a wff and furthermore $\beta = a_i a_{i+1} \ldots a_j$ for some i, j with $i + j \leq n$.

Lemma: Properties of Subformulas. Let β, γ, δ be nonempty strings. It cannot happen that $\beta\gamma$ and $\gamma\delta$ are both wffs, as otherwise γ would both to have more closing brackets than opening brackets and more opening brackets than closing brackets.

Furthermore, if β is a wff then $\beta\gamma$ cannot be a wff. Thus one can conclude that if β, γ are subformulas of α then either they occur at disjoint places in α or β is a subformula of γ or γ is a subformula of β.

If α is a wff and neither an atom nor a truth-value then there are subformulas β, γ of α such that either $\alpha = (\neg\beta)$ and $\gamma = \beta$ or $\alpha = (\beta \wedge \gamma)$ or $\alpha = (\beta \vee \gamma)$ or $\alpha = (\beta \oplus \gamma)$ or $\alpha = (\beta \rightarrow \gamma)$ or $\alpha = (\beta \leftrightarrow \gamma)$. Furthermore the subformulas β, γ are maximal, that is, there is no subformula δ of α which contains one of β and γ as a subformula. It can also be seen that the choice of the subformulas β, γ and the connective in the above is unique.

Proposition. Every construction sequence for a wff α contains besides α all subformulas of α.

Proof. This is clearly true when α is an atom or a truth-value, as α does not have subformulas. Assume that this is true for β and γ. Then a construction sequence for $\alpha = (\beta \rightarrow \gamma)$ must contain both β and γ due to the fact that there is only one way to obtain α as the output of a constructor function and β, γ are the corresponding inputs. Furthermore, the construction sequence must also be one for β and γ, thus by induction hypothesis, all subformulas of β and γ appear in the construction sequence. From the form of α and the preceding properties of the subformulas, one can see that there are no further subformulas of α then β, γ and their subformulas. The same can be shown for the connectives $\leftrightarrow, \vee, \wedge, \oplus$ in place of $\rightarrow$ in the above. Similarly, if $\alpha = (\neg\beta)$ then again every subformula of α is either β or a subformula of β. Due to the uniqueness of the decomposition, there is only one way on how to write α and β must appear in the construction sequence. Furthermore, by induction hypothesis, all subformulas of β also occur there. $\square$

Recursion Theorem. Assume that C is freely generated subset of D with respect to a base-set B and constructor functions f, g and further assume that the "generation is free", that is, for each $x \in C$, exactly one of the following cases holds:

- $x \in B$;
- There are $y, z \in C$ with $x = f(y, z)$ and y, z uniquely depend on x;
- There is an $y \in C$ with $x = g(y)$ and y uniquely depends on x.

Furthermore, assume that there is a further set E and that there are functions $h : B \to E$, $\overline{f} : E \times E \to E$ and $\overline{g} : E \to E$. Then there is a unique function $\overline{h}$ such that the following holds:

1. For all $x \in B$, $\overline{h}(x) = h(x)$;
2. For all $x, y \in B$, $\overline{h}(f(x,y)) = \overline{f}(\overline{h}(x), \overline{h}(y))$;
3. For all $x \in C$, $\overline{h}(g(x)) = \overline{g}(\overline{h}(x))$.

Proof Idea. One proves by induction that the function $\overline{h}$ is defined on the whole of C and that it is unique.

First, for all $x \in B$, $\overline{h}(x) = h(x)$ by the above definition and none of the other cases applies, so $\overline{h}$ is defined and unique there.

Second, assume that $x = f(y,z)$ and the statement is already proven for y and z by induction hypothesis. Then $x \notin B$ and furtherore, $\overline{h}(y), \overline{h}(z)$ are already uniquely defined. Now there is no other way to obtain x from other elements using f, g and B and therefore, $\overline{h}(x) = \overline{f}(\overline{h}(y), \overline{h}(z))$ is the only possible definition and this definition also applies, hence $\overline{h}(x)$ is defined and unique.

Third, assume that $x = g(y)$ and the statement is already proven for y by induction hypothesis. Then $x \notin B$ and furthermore, $\overline{h}(y)$ is already uniquely defined. Now there is no other way to obtain x from other elements using f, g and B and therefore, $\overline{h}(x) = \overline{g}(\overline{h}(y))$ is the only possible definition and this definition also applies, hence $\overline{h}(x)$ is defined and unique.

Proof. One defines K to be the set of all partial functions h' from C to E which satisfy the following:

1. If $x \in B$ and $h'(x)$ is defined then $h'(x) = h(x)$;
2. If $x = f(y,z)$ and $h'(x)$ is defined, then $h'(y)$ and $h'(z)$ are also defined and $h'(x) = \overline{f}(h'(y), h'(z))$;
3. If $x = g(y)$ and $h'(x)$ is defined, then $h'(y)$ is also defined and $h'(x) = \overline{g}(h'(y))$.

Now one shows the following two conditions:

(a) For every $x \in C$ there is a $h' \in K$ with $h'(x)$ being defined;

(b) For every $x \in C$ and all $h', h'' \in K$, if $h'(x)$ and $h''(x)$ are both defined then they are equal.

First one shows that for every constructor sequence $s = (x_1, x_2, \ldots, x_n)$ there exists a function $h_s \in K$ which satisfies the above two conditions. Without loss of generality, one can assume that s is repetition-free, that is, no element equals to x_i and x_j in s for two different i and j. has two indices in s. As one has only finitely many values, one can choose these explicitly as follows for $m = 1, 2, \ldots, n$:

1. If $x_m \in B$ then one chooses $h_s(x_m) = h(x_m)$;
2. If $x_m = f(x_i, x_j)$ for $i, j < m$ then one chooses $h_s(x_m) = \overline{f}(h_s(x_i), h_s(x_j))$;
3. If $x_m = g(x_k)$ for $k < m$ then one chooses $h_s(x_m) = \overline{g}(h_s(x_k))$.

The inductive construction of h_s enforces that $h_s \in K$ and so there is a member of K on which every member of the construction sequence is defined. Thus condition (a) is satisfied.

For condition (b), one considers the following set S: S is the set of all $x \in C$ such that there do not exist h', h'' which are defined and different on x. One shows that the set S is inductive.

1. If $x \in B$ then $x \in S$, as all members $h', h'' \in K$ which are defined on x satisfy that $h'(x) = h(x)$ and $h''(x) = h(x)$ and thus h' and h'' are equal.
2. If $x = f(y, z)$ then there is no alternative way to obtain x from other elements and $x \notin B$. Assume now that $h', h'' \in K$ satisfy that $h'(x)$ and $h''(x)$ are defined. Then also $h'(y), h''(y), h'(z), h''(z)$ are all defined due to membership of h', h'' in K. By induction hypothesis, $h'(y) = h''(y)$ and $h'(z) = h''(z)$. Now
$$h'(x) = \overline{f}(h'(y), h'(z)) = \overline{f}(h''(y), h''(z)) = h''(x).$$
3. If $x = g(y)$ then there is no alternative way to obtain x from other elements of C and $x \notin B$. Assume now that that $h', h'' \in K$ satisfy that $h'(x)$ and $h''(x)$ are defined. Then also $h'(y)$ and $h''(y)$ are defined and by induction hypothesis, $h'(y) = h''(y)$.

Thus h' and h'' coincide on x and $x \in S$. Thus one can define $\overline{h}(x)$ for any $x \in C$ as that value which is $h'(x)$ for some $h' \in K$ and by assumption, this value is unique. So $\overline{h}$ exists by the rules of set theory (the more direct proof idea does not give this). □

Chapter 1.5 – Sentential Connectives

Additional Connectives. One could introduce another connective, called # or majority. As this connective has three inputs, one writes it as a function with arguments separated by commas: $(\#(\alpha, \beta, \gamma))$; here the outer brackets can be omitted in informal writing. The interpretation of this connective is that when α, β, γ have values from 0 and 1, that the connective follows the majority-vote on whether it should be 1 or 0. A question is whether such a connective is something new. Indeed, it can be expressed by the other connectives in several ways:

$$\begin{aligned} \#(\alpha, \beta, \gamma) &= (\alpha \wedge \beta) \vee (\alpha \wedge \gamma) \vee (\beta \wedge \gamma); \\ \#(\alpha, \beta, \gamma) &= (\alpha \vee \beta) \wedge (\alpha \vee \gamma) \wedge (\beta \vee \gamma). \end{aligned}$$

One could view this also as functions which map $\{0,1\}^3$ to $\{0,1\}$ and in the text which follows, the notion of a Boolean function which maps inputs from $\{0,1\}^n$ to $\{0,1\}$; the constants 0 and 1 can also be viewed as Boolean functions which map nullary input to an output which is always the same. Then $\neg$ can be viewed as a Boolean function from $\{0,1\}$ to $\{0,1\}$ which maps 0 to 1 and 1 to 0. The other connectives are Boolean functions from $\{0,1\}^2$ to $\{0,1\}$.

Boolean Functions defined by wffs. For a wff α using atoms $A_1, A_2, \ldots, A_n$, one can define the Boolean function $B^n_\alpha(x_1, \ldots, x_n)$ as the function which is tabled down by the truth-table for the formula α. For example, $B_{A_1 \wedge A_2}$ is given by the following table:

A_1	A_2	$\alpha = A_1 \wedge A_2$	B_α
0	0	0	$B_\alpha(0,0) = 0$
0	1	0	$B_\alpha(0,1) = 0$
1	0	0	$B_\alpha(1,0) = 0$
1	1	1	$B_\alpha(1,1) = 1$

Note that one also has Boolean functions which one normally does not explicitly be aware of like projections. Examples are

$$B^3_{A_2}(x_1, x_2, x_3) = x_2 \text{ and } B^4_{A_2 \wedge A_3}(x_1, x_2, x_3, x_4) = x_2 \wedge x_3.$$

Given two wff α and β, one defines that $B^n_\alpha \leq B^n_\beta$ iff for all $(x_1, \ldots, x_n) \in \{0,1\}^n$, $B^n_\alpha(x_1, \ldots, x_n) \leq B^n_\beta(x_1, \ldots, x_n)$, where $0 \leq 1$. This gives rise to the following theorem.

Theorem 15A. Let α and β be wffs whose sentence symbols are among $A_1, \ldots, A_n$. Then the following statements are true:

(a) $\{\alpha\} \models \beta$ iff $B^n_\alpha \leq B^n_\beta$;

(b) $\{\alpha\} \models \beta$ and $\{\beta\} \models \alpha$ (that is, α and β are equivalent) iff $B^n_\alpha = B^n_\beta$;

(c) $\emptyset \models \alpha$ iff $B^n_\alpha = B^n_1$, that is, iff B^n_α is the n-input function which always outputs 1.

Proof. (a) If $\{\alpha\} \models \beta$ then for every ν with $\overline{\nu}(\alpha) = 1$, it also holds that $\overline{\nu}(\beta) = 1$. This is equivalent to saying that $\overline{\nu}(\alpha) \leq \overline{\nu}(\beta)$.

For every $x_1, \ldots, x_n \in \{0,1\}^n$ there is a ν with $\nu(A_1) = x_1, \ldots, \nu(A_n) = x_n$, it follows that $B_\alpha(x_1, \ldots, x_n) = \overline{\nu}(\alpha) \leq \overline{\nu}(\beta) = B_\beta(x_1, \ldots, x_n)$. This gives one direction of the equivalence.

Furthermore, if for every $x_1, \ldots, x_n$ it holds that $B_\alpha(x_1, \ldots, x_n) \leq B_\beta(x_1, \ldots, x_n)$ and ν is any truth-assginment, then let $x_m = \nu(A_m)$ for $m = 1, 2, \ldots, n$ and $\overline{\nu}(\alpha) = B_\alpha(x_1, \ldots, x_n) \leq B_\beta(x_1, \ldots, x_n) = \overline{\nu}(\beta)$. As this holds for all ν, $\{\alpha\} \models \beta$. This gives the second direction of the equivalence.

(b) α and β are equal as functions iff, for all $x_1, \ldots, x_n$, the equality $B_\alpha(x_1, \ldots, x_n) = B_\beta(x_1, \ldots, x_n)$ holds. It is easy to see that this equality is the same as saying both $B_\alpha \leq B_\beta$ and $B_\beta \leq B_\alpha$ hold. Thus part (b) follows directly from part (a).

(c) This part follows from the fact that a formula is a tautology iff it is equivalent to the formula 1. Note that the formula 1 is the easiest formula which is true irrespective of the values of the truth-values of the atoms. If one does not want to use truth-values, the formula $(A_1 \vee (\neg A_1))$ would also do. $\square$

Theorem 15B. For every n-place Boolean function f there is a wff α using atoms $A_1, \ldots, A_n$ such that $f = B^n_\alpha$.

Proof. In the following, let 0 be the value of an empty disjunction of formulas. That is, if one makes a disjunction for which no terms qualify then one lets the resulting formula just be 0.

The idea of the proof the following: Given an n-ary function $f(x_1, \ldots, x_n)$, one computes for each $(x_1, \ldots, x_n) \in \{0,1\}^n$ the value $f(x_1, \ldots, x_n)$. Now for each $(x_1, \ldots, x_n) \in \{0,1\}^n$ let $\alpha_{x_1,\ldots,x_n}$ denote the conjunction of all A_m with $x_m = 1$ and $(\neg A_m)$ with $x_m = 0$; so $\alpha_{0,1,1,0}$ is $(\neg A_1) \wedge A_2 \wedge A_3 \wedge (\neg A_4)$ when omitting brackets for readability. Now one lets α to be the disjunction of all $\alpha_{x_1,\ldots,x_n}$ where

$f(x_1, \ldots, x_n) = 1$. Note that when $(x_1, \ldots, x_n)$ and $(y_1, \ldots, y_n)$ are two different tuples from $\{0,1\}^n$ then there is no ν which makes $\overline{\nu}(\alpha_{x_1,\ldots,x_n})$ and $\overline{\nu}(\alpha_{y_1,\ldots,y_n})$ true at the same time: The reason is that there is one m with $x_m \neq y_m$, say $x_m = 1$ and $y_m = 0$. Then $\overline{\nu}(\alpha_{x_1,\ldots,x_n}) = 1$ only if $A_m = 1$ and $\overline{\nu}(\alpha_{y_1,\ldots,y_n}) = 1$ only if $A_m = 0$ and both does not happen at the same time.

Recall that $B_\alpha(x_1, \ldots, x_n)$ is the value $\overline{\nu}(\alpha)$ for that ν which satisfies $\nu(A_1) = x_1, \nu(A_2) = x_2, \ldots, \nu(A_n) = x_n$. If $f(x_1, \ldots, x_n) = 1$ then $\alpha_{x_1,\ldots,x_n}$ occurs in the disjunction of formulas put together for B_α and therefore $B_{\alpha_{x_1,\ldots,x_n}}(x_1, \ldots, x_n) = 1$ and $B_\alpha(x_1, \ldots, x_n) = 1$. If $f(x_1, \ldots, x_n) = 0$ then $\alpha_{x_1,\ldots,x_n}$ does not occur in the disjunction of formulas forming B_α and all the $\alpha_{y_1,\ldots,y_n}$ occurring in the disjunction satisfy that $B_{\alpha_{y_1,\ldots,y_n}}(x_1, \ldots, x_n) = 0$ and therefore also $B_\alpha(x_1, \ldots, x_n) = 0$. Thus $B_\alpha(x_1, \ldots, x_n) = f(x_1, \ldots, x_n)$ for all $(x_1, \ldots, x_n) \in \{0,1\}^n$. $\square$

Example. Consider the function f given by the following truth-table:

x_1	x_2	x_3	$f(x_1, x_2, x_3)$
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

This formula evaluates to 1 iff either all three inputs are 0 or all three inputs are 1. One can therefore represent the formula as the disjunction of the two disjunctions which tests that all atoms are 0 or that all atoms are 1, which is done as follows:

$$\alpha = ((\neg A_1) \wedge (\neg A_2) \wedge (\neg A_3)) \vee (A_1 \wedge A_2 \wedge A_3).$$

So this formula realises the function which checks whether all inputs are equal.

Normalforms. A literal of a formula is either an atom or a negated atom. A conjunctive clause is a conjunction of literals; a (disjunctive) clause is a disjunction of literals. There are several normal forms of Boolean formulas. The disjunctive normalform is a disjunction of conjunctive clauses; the conjunctive normalform is a

conjunction of disjunctive clauses, the latter are also often referred to as "clauses". The proof of Theorem 15B gives also rise to the following corollary.

Corollary 15C. For every α one can find an equivalent β in disjunctive normal form.

Completeness. A set S of connectives is complete if all Boolean functions with at least one input variable can be represented by a formula α using these connectives, that is, for each $n \geq 1$ and each n-ary Boolean function f there is a wff α using only the first n atoms and the connectives just mentions such that for all $(x_1, \ldots, x_n) \in \{0,1\}^n$ the equality $B_\alpha(x_1, \ldots, x_n) = f(x_1, \ldots, x_n)$ holds. The proof of Theorem 15B enable to prove also the following result.

Theorem 15D. The following sets of connectives are complete: $\{\neg, \wedge, \vee\}$, $\{\neg, \wedge\}$ and $\{\neg, \vee\}$.

Proof. For the case of $\{\neg, \wedge, \vee\}$, one can directly invoke Theorem 15B. For the case of $\neg, \wedge$, one uses that $\beta \vee \gamma$ is equivalent to $\neg((\neg\beta) \wedge (\neg\gamma))$ by De Morgan's Law. Similarly, the same laws show that $\beta \wedge \gamma$ is equivalent to $\neg((\neg\beta) \vee (\neg\gamma))$. Thus one can from each formula eliminate one of $\wedge$ and $\vee$ and systematically replace it by the other one. Here an example: The formula

$$(A_1 \wedge (A_2 \vee A_3))$$

can be written in the following two ways:

$$(\neg((\neg A_1) \vee (\neg(A_2 \vee A_3)))) \text{ and } (A_1 \wedge (\neg((\neg A_2) \wedge (\neg A_3)))).$$

The same can be done iteratively with more complicated formulas. That is, when eliminating $\vee$ one replaces each subformula of the form $(\beta \vee \gamma)$ by $(\neg((\neg\beta) \wedge (\neg\gamma)))$ until no replacement of that form can be done. When eliminating $\wedge$, one replaces every subformula of the form $(\beta \wedge \gamma)$ by $(\neg((\neg\beta) \vee (\neg\gamma)))$ until no replacement of that form can be done. $\square$

Example. In some cases, one needs the truth-values to get completeness. For example, $\{0, 1, \rightarrow\}$ is complete: One can replace $\neg\beta$ by $(\beta \rightarrow 0)$: If $\overline{\nu}(\beta) = 1$ then $\overline{\nu}((\beta \rightarrow 0)) = 0$ and if $\overline{\nu}(\beta) = 0$ then $\overline{\nu}((\beta \rightarrow 0)) = 1$. Furthermore, $\overline{\nu}((\beta \vee \gamma)) = \overline{\nu}(((\neg\beta) \rightarrow \gamma))$, so that both $\neg$ and $\vee$ can be expressed using $0, 1, \rightarrow$.

However, without the truth-values, it is impossible to express the negation, therefore even $\{\rightarrow, \leftrightarrow, \wedge, \vee\}$ is incomplete. To see this, one shows the following: If $\nu(A_m) =$

1 for all atoms A_m, then $\overline{\nu}(\alpha) = 1$ for all formulas α which are obtained from the atoms in finitely many steps by using only connectives from $\rightarrow, \leftrightarrow, \wedge, \vee$: The induction step is that if β, γ are fomulas with $\overline{\nu}(\beta) = 1$ and $\overline{\nu}(\gamma) = 1$, so are $\overline{\nu}((\beta \rightarrow \gamma))$, $\overline{\nu}((\beta \leftrightarrow \gamma))$, $\overline{\nu}((\beta \wedge \gamma))$ and $\overline{\nu}((\beta \vee \gamma))$.

Nullary and Unary Connectives. The truth-values 0 and 1 are the only nullary connectives. However, if α is a tautology with n atoms, then B_α is an n-ary connective of the same value as 1. Furthermore, $B_{\neg\alpha}$ is a n-ary connective of the value 0. Thus, out of the four unary connectives, two are already the two constant functions. The other two are the identity function B_{A_1} and the negation $B_{\neg A_1}$.

Binary Connectives and Ternary Connectives. There are sixteen binary connectives, out of which six are essential unary or nullary. The connectives are given in this table.

0	1	truth-values
A_1 $\neg A_1$	A_2 $\neg A_2$	copying one input negating one input
$A_1 \wedge A_2$ $\neg(A_1 \wedge A_2)$	$A_1 \vee A_2$ $\neg(A_1 \vee A_2)$	and, or nand, nor
$A_1 \rightarrow A_2$ $(\neg A_1) \wedge A_2$	$A_1 \leftarrow A_2$ $A_1 \wedge (\neg A_2)$	implications negated implications
$A_1 \leftrightarrow A_2$	$A_1 \oplus A_2$	equivalence, exclusive or

The "nand" is sometimes written by a vertical stroke, the Sheffer stroke; however, the vertical stroke is used in some programming languages just to denote the "or", so it is a bit ambiguous. The exclusive or is also just the addition in the binary field (odd plus odd is even) where and is the multiplication.

There are 256 ternary connectives, out of which many are essentially nullary, unary or binary, that is, do not depend on all variables. Furthermore, as there are so many of them, many of these connectives do not have a proper name. One of these showed up before, namely the majority-connective.

Examples. The sets {nand} and {nor} are both complete. The reason is given by the following table of and, or and not:

operation	with nand	with nor
$\neg\alpha$	nand(α, α)	nor(α, α)
$\alpha \wedge \beta$	nand(nand(α, β), nand(α, β))	nor(nor(α, α), nor(β, β))
$\alpha \vee \beta$	nand(nand(α, α), nand(β, β))	nor(nor(α, β), nor(α, β))

The idea is, depending on what is needed, either to negate the output of nand / nor in order to recover the and / or from it or to negate the input in order to apply De Morgan's Law. The negation is done by supplying the same input twice, so that the disjunction or conjunction before the negation is filtered out and only the negation is done. This then permits to implement both the negation of the output of a nand / nor as well as the negation of the inputs for the De Morgan's Law. □

One Sample-Definition of Fuzzy Logic. Fuzzy logic is a multi-valued logic, where the truth-values are linearly ordered. For this, one let the truth-values be a nonempty set $Q \subseteq \mathbb{Q}$ of rationals where $0 = \min(Q)$, $1 = \max(Q)$ and Q has the following closure properties: If $p, q \in Q$ so are also $1 - p$ and $\min\{p + q, 2 - p - q\}$. Thus Q contains $0, 1$ (as the set has a minimum and maximum of these values) and if $p+q \leq 1$ then $p+q \in Q$ and if $p+q \geq 1$ then $p+q-1 = 1-(2-p-q) \in Q$. Clearly $Q = \{0, 1\}$ is possible, but also $Q = \{0.00, 0.01, \ldots, 0.99, 1.00\}$ and $Q = \{q \in \mathbb{Q} : 0 \leq q \leq 1\}$. Now one can define the following connectives on truth-values using $\min, \max, +, -$ and constants form $\mathbb{Q}$ in the same way as for rational numbers:

$$\begin{aligned} p \wedge q &= \min\{p, q\}; \\ p \vee q &= \max\{p, q\}; \\ \neg p &= 1 - p; \\ p \to q &= \min\{1 + q - p, 1\}; \\ p \leftrightarrow q &= \min\{1 + p - q, 1 + q - p\}; \\ p \oplus q &= \min\{p + q, 2 - p - q\}. \end{aligned}$$

Given now a truth-assignment with $\nu(A_k) \in Q$ for all k, one can extend ν to $\overline{\nu}$ by induction:

1. $\overline{\nu}(A_k) = \nu(A_k)$ and $\overline{\nu}(p) = p$ for all $p \in Q$;
2. $\overline{\nu}(\neg(\alpha)) = 1 - \overline{\nu}(\alpha)$;
3. $\overline{\nu}(\alpha \vee \beta) = \max\{\overline{\nu}(\alpha), \overline{\nu}(\beta)\}$ and $\overline{\nu}(\alpha \wedge \beta) = \min\{\overline{\nu}(\alpha), \overline{\nu}(\beta)\}$ and similarly for all other binary connectives following the formulas above.

One can define additional connectives, where some do not exist for all Q:

1. $med(p, q, r)$ is the middle value (median) of the three truth-values p, q, r; it is given by the same formula as the majority which is $med(p, q, r) = (p \wedge q) \vee (p \wedge r) \vee (q \wedge r)$;

2. $ave(p,q,r) = (p+q+r)/3$ which only exists when Q is closed under the mapping $p,q,r \mapsto (p+q+r)/3$ and this can only happen when Q is infinite.

To see that the two are different, note that $med(0,1,1) = 1$ and $ave(0,1,1) = 2/3$. Furthermore, only some but not all tautologies from normal logic become tautologies in fuzzy logic. For this one has first to formalise tautologies:

1. One says that $S \models \alpha$ iff for every ν there is a $\beta \in S \cup \{1\}$ with $\overline{\nu}(\beta) \leq \overline{\nu}(\alpha)$;

2. Now α is a tautology iff $\emptyset \models \alpha$;

3. All formulas made from atoms and connectives $\wedge, \vee, \neg$ are evaluated to $1/2$ in the case that $1/2 \in Q$ and all atoms take the value $1/2$ and hence none of them is a tautology;

4. However, $\alpha \leftrightarrow \alpha$ and $\alpha \rightarrow \alpha$ are tautologies and indeed, $\alpha \leftrightarrow \beta$ is a tautology iff for all Q-valued truth-assignments ν, $\overline{\nu}(\alpha) = \overline{\nu}(\beta)$.

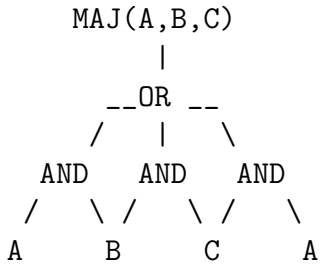
Note that the exact formulation of "fuzzy logic" is not fixed: There are various versions of it around which differ in the choice of Q and which also differ in the choice of the formulas defining the connectives. Furthermore, besides $S \models \alpha$, one might also be interested in $S \models_p \alpha$ meaning that for every ν there is a $\beta \in S \cup \{1\}$ with $\overline{\nu}(\beta) + p - 1 \leq \overline{\nu}(\alpha)$. So $\emptyset \models_{0.8} \alpha$ would only require the following:

$$\forall Q\text{-valued truth-assignments } \nu\, [\overline{\nu}(\alpha) \geq 0.8].$$

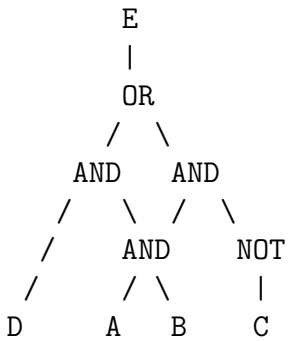
The idea behind it is that the truth-values might be associated with amount of certainty that something is true: 0 is false, 0.2 is unlikely, 0.5 is unknown, 0.8 is likely, 1 is true. Then one might put into S rules like "A_4 is likely" which is expressed as "$0.8 \rightarrow A_4$" and so on. After having done this, one wants know what one can say about α at the end; now $S \models_{0.8} \alpha$ would mean that for all truth-assignments, S likely implies α.

Chapter 1.6 – Switching Circuits

For better graphics, please see the textbook. Circuits can be seen as inputs mapped to outputs using some basic connectives; the main idea of circuits is that one, other than in formulas, can reuse inputs into several outputs. Furthermore, one might have gates with more than two inputs, in particular for AND and OR it is common to allow as many inputs as needed. Here an example for the majority functions, for simpler drawing, the input A is duplicated.

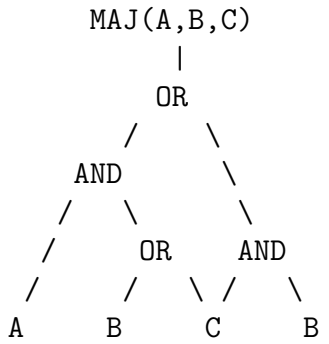


The single units in a circuit are called a gate and gates might cause delays, for example each OR might have a processing time of 1 time unit to adjust the output to the input and again the AND has a further processing time of another time unit, giving an overall processing time of two time units. One goals of the layouts is to keep such a delay to a minimum. Sometimes, formulas go over more levels, as in the following example, which has a circuit computing $E = ((A \wedge B) \wedge D) \vee ((A \wedge B) \wedge \neg C)$ where the subpart $A \wedge B$ has two outputs to avoid duplication in the layout.



The number of layers is called the depth of the circuit. Here there are three layers of gates, each causing one delay. If one does not have a multiple input for the above

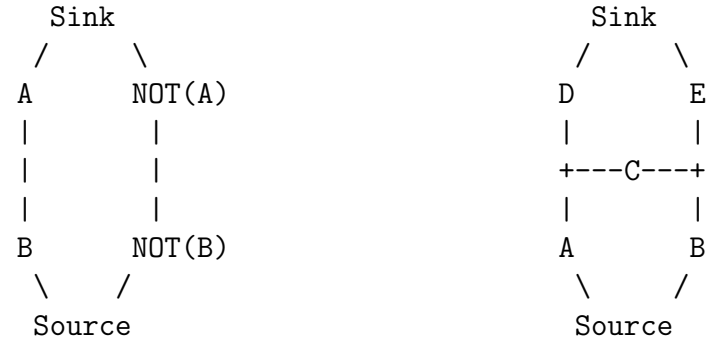
majority circuit, one also needs three layers; however, one can bring down from five to four gates.



Again, for better drawing, the input B is duplicated.

Quiz. The textbook gives a circuit for $\leftrightarrow$ made of NOR-gates. Make one with NAND-gates and NOT-gates only. Note that NOT-gates use less inputs than NAND-gates, but can be simulated by the latter with input-duplication.

Relay Switching. Idea is to use the inputs as relays: When they are 1, the current can flow, when they are 0, it cannot flow. Now the circuit is 1 iff there is a flow from the source to the sink. Costs are usages of inputs and not the wires. Inputs can be negated. Here two examples.



The first circuit is for $(A \wedge B) \vee (\neg A \wedge \neg B)$; the second for $(A \wedge D) \vee (A \wedge C \wedge E) \vee (B \wedge C \wedge D) \vee (B \wedge E)$. The surprising usage of input C is that it might allow flows in either direction.

Incomplete Specifications. One might also ask what how to realise a formula or circuit efficiently in the case that it is underspecified; that is, how to choose the missing values in the truth-table such that the formula gets as easy as possible. For four variables, Enderton gives the formula as a square-diagramme, for example like this:

	NOT(A)	NOT(A)	A	A	
NOT(C)	1	.	0	.	NOT(D)
NOT(C)	1	1	0	1	D
C	.	1	0	1	D
C	0	.	0	0	NOT(D)
	NOT(B)	B	B	NOT(B)	

Quiz. Now the task is to fill the dots such that one can solve the formula with three binary AND and OR gates and can use as many NOT gates as one wants. What solution is proposed?

Chapter 1.7 – Compactness and Effectiveness

The compactness theorem deals with the notion of satisfiability. Here one says that a set of formulas S is satisfiable iff there is a truth-assignment ν such that, for all $\alpha \in S$, $\overline{\nu}(\alpha) = 1$.

Quiz. Which of the following sets of formulas are satisfiable:

1. $\{A_1, A_2, \neg A_1 \vee \neg A_2\}$;
2. $\{A_1 \vee \neg A_2, A_2 \vee \neg A_3, A_3 \vee \neg A_1\}$;
3. $\{\neg((A_1 \wedge A_2) \leftrightarrow (A_2 \wedge A_1))\}$.

Compactness Theorem. A set S of wffs is satisfiable iff every finite subset is satisfiable.

Proof. For making the proof notationally easier, one says that a set S *is finitely satisfiable* iff every finite subset of S is satisfiable. So one has to prove that S is satisfiable iff S is finitely satisfiable and the implication "S is satisfiable $\Rightarrow$ S is finitely satisfiable" follows directly from the fact that every subset of a satisfiable set is satisfiable. So assume that S is finitely satisfiable. Furthermore, one can assume that S is infinite, as otherwise there is nothing to prove. Let $\alpha_1, \alpha_2, \ldots$ be an enumeration of all wffs which exist.

Now one constructs a supserset T of S such that T contains for each formula α_k a truth-value $q_k \in \{0, 1\}$ such that $\alpha \leftrightarrow q$ is in T. This is done inductively. So assume that one has defined all q_m for $m < n$ and the set $S_n = S \cup \{\alpha_m \leftrightarrow q_m : 1 \leq m \leq n\}$ is finitely satisfiable; note that $S_0 = S$. Now there are for each $n \geq 0$ two cases:

- There is for each truth-value $q \in \{0, 1\}$ a finite subset R_q of S_n such that $R_q \cup \{\alpha_{n+1} \leftrightarrow q\}$ is unsatisfiable;
- There is a truth-value $q_{n+1} \in \{0, 1\}$ such that $S_n \cup \{\alpha_{n+1} \leftrightarrow q_{n+1}\}$ is finitely satisfiable and one chooses $S_{n+1} = S_n \cup \{\alpha_{n+1} \leftrightarrow q_{n+1}\}$ in the inductive step accordingly (to break ties, q_{n+1} is the minimal truth-value which can be chosen here).

So in the first case the inductive construction would get stuck and in the second case it would go through. Thus one has to prove that the first case never happens. So assume that the sets R_0 and R_1 would exist. Then one considers the set $R_0 \cup R_1$. This set is a finite subset of S_n and thus satisfiable by a truth-assignment ν. Now there is a truth-value $q = \overline{\nu}(\alpha_{n+1})$. Now one determines the truth-value $\overline{\nu}(\alpha_{n+1} \leftrightarrow q)$. This is equal to the truth-value of $\overline{\nu}(\alpha_{n+1}) \leftrightarrow q$ and thus to $q \leftrightarrow q$. This evaluates to 1. As $R_0 \cup R_1 \cup \{\alpha_{n+1} \leftrightarrow q\}$ is satisfied by ν, ν also satisfies the subset $R_q \cup \{\alpha_{n+1} \leftrightarrow q\}$. This contradicts the assumption, thus Case 1 does not hold.

Thus the inductive contruction gives rise to an ascending family S_n of finitely satisfiable sets and their union T satisfies that every finite subset of T is actually a finite subset of some S_n; hence T is also finitely satisfiable. In order to simplify notation, one defines q_α to be q_k for that index k which satisfies $\alpha_k = \alpha$.

Now one defines for each atom A_m that $\mu(A_m) = q_{A_m}$ and it remains to show all α satisfy $\overline{\mu}(\alpha) = q_\alpha$. This is done by structural induction; for the atoms it follows by definition. Furthermore, for all truth-values p, it holds that the formula $q_p \leftrightarrow p$ is a tautoloy when $q_p = p$ and unsatisfiable when $q_p \neq p$; thus $q_p = p$ for $p = 0, 1$ and consequently $\overline{\mu}(p) = q_p$. For the inductive step, consider any α, β for which it is already verified that $\overline{\mu}(\alpha) = q_\alpha$ and $\overline{\mu}(\beta) = q_\beta$.

For $\neg\alpha$, one has that $\overline{\mu}(\neg\alpha) = \neg q_\alpha$ by induction hypothesis and thus $\overline{\mu}(\neg\alpha \leftrightarrow \neg q_\alpha) = 1$. Furthermore, as T is finitely satisfiable, the finite set $\{\alpha \leftrightarrow q_\alpha, \neg\alpha \leftrightarrow q_{\neg\alpha}\}$ is satisfiable and therefore also the set $\{\alpha \leftrightarrow q_\alpha, \neg\alpha \leftrightarrow \neg q_\alpha, \neg\alpha \leftrightarrow q_{\neg\alpha}\}$ is satisfiable, as $p \leftrightarrow q$ implies $\neg p \leftrightarrow \neg q$ for all truth-values. Now a set containing $\alpha \leftrightarrow p$ and $\alpha \leftrightarrow q$ for expressions p, q in truth-values is only satisfiable when p, q represent the same truth-value. Thus $\neg q_\alpha$ and $q_{\neg\alpha}$ are the same truth-value. Furthermore, $\overline{\mu}(\neg\alpha) = \neg\overline{\mu}(\alpha) = \neg q_\alpha$ by induction hypothesis and therefore $q_{\neg\alpha} = \overline{\mu}(\neg\alpha)$.

Now note that similarly, for a truth-value p, the set $\{\alpha \leftrightarrow q_\alpha, \beta \leftrightarrow q_\beta, \alpha \wedge \beta \leftrightarrow p\}$ is satisfiable iff $p = q_\alpha \wedge q_\beta$. Thus one can conclude that $q_{\alpha\wedge\beta} = q_\alpha \wedge q_\beta$ and therefore $\overline{\mu}(\alpha \wedge \beta) = \overline{\mu}(\alpha) \wedge \overline{\mu}(\beta) = q_\alpha \wedge q_\beta = q_{\alpha\wedge\beta}$. The same arguments hold for the other connectives $\vee, \oplus, \leftrightarrow, \neg$. This induction completes the proof. $\square$

Corollary 17A. If $S \models \alpha$ then there is a finite subset S' of S with $S \models \alpha$.

Proof. One uses the basic fact that $S \models \alpha$ iff $S \cup \{\neg\alpha\}$ is unsatisfiable. If now every finite $S' \subseteq S$ satisfies $S' \not\models \alpha$ then $S' \cup \{\neg\alpha\}$ is satisfiable for every finite $S' \subseteq S$ and $S \cup \{\neg\alpha\}$ is finitely satisfiable. This implies that $S \cup \{\neg\alpha\}$ is satisfiable and $S \not\models \alpha$. $\square$

Compactness Theorem and Fuzzy Logic. The above proof of the Compactness

Theorem works also for fuzzy logics where Q is a finite set, one has of course then to invoke the formulas for $\leftrightarrow$ and the other connectives given at the end of Chapter 1.5. However, the proof does not work if the set Q is that of all rationals between 0 and 1. The reason is that for this infinite set, the result is false.

The counter example is the following one: Let r be an irrational number with $0 < r < 1$, say $r = \sqrt{1/2}$. Furthermore, let S contain for every $q \in Q$ the following formula α_q: If $q < r$ then α_q is $q \to A_1$; if $q > r$ then α_q is $A_1 \to q$. Note that $r \notin Q$ and therefore the case $q = r$ does not occur. Now S is finitely satisfiable; if $R \subseteq S$ is finite then there is some $r' \in Q$ such that all $\alpha_q \in R$ satisfy that if $q \leq r'$ then $\alpha_q = q \to A_1$ else $\alpha_q = A_1 \to q$. Now taking $\nu(A_1) = r'$ makes all $\alpha_q \in R$ true, as $q \to r'$ evaluates to 1 for all $q \leq r'$ and $r' \to q$ evaluates to 1 for all $q > r'$. However, S is not satisfiable, as there is no possible truth-value $r' \in Q$ which A_1 can take so that for all rationals q, $q < r'$ iff $q < r$. This proof can be adjusted to any situation where Q is a dense subset of $\{r \in \mathbb{R} : 0 \leq r \leq 1\}$ which does not contain all the reals r between 0 and 1. Thus the best what one can hope for is to show the compactness theorem for the cases of $Q = \{0, 1/k, 2/k, \ldots, (k-1)/k, 1\}$ where k is a natural number with $k \geq 1$ and $Q = \{r \in \mathbb{R} : 0 \leq r \leq 1\}$. The first case is already covered by above the proof for the compactness theorem; for the second case, one needs some preparation and the resulting theorem would also work for the first case, including the classical case of $k = 1$.

Continuity of the Functions $\mathbf{B}_\alpha$. In the case of a finite Q, one equips it with the discrete topology and all functions from Q^n to Q are continuous. In the case of $Q = \{r \in \mathbb{R} : 0 \leq r \leq 1\}$, a close inspection of the formulas for the connectives shows that each of them are continuous functions. Furthermore, constant functions B^n_q which maps n inputs to a constant $q \in Q$ is continuous and the function which maps the m-th input to the output, that is, $B^n_{A_m}$, is also continuous. Now, for the inductive step, one can given two continuous functions B^n_α and B^n_β see that the combined functions

$$\begin{aligned} B^n_{\neg\alpha}(x_1, \ldots, x_n) &= 1 - B^n_\alpha(x_1, \ldots, x_n) \text{ and} \\ B_{\alpha\wedge\beta}(x_1, \ldots, x_n) &= \min\{B_\alpha(x_1, \ldots, x_n), B_\beta(x_1, \ldots, x_n)\} \end{aligned}$$

as well as the combined functions for the other connectives are concatenations of continuous functions and thus continuous.

Compactness Theorem for Fuzzy Logic. Let Q be a compact set of possible truth-values for fuzzy logic, that is, either $Q = \{0, 1/k, 2/k, \ldots, (k-1)/k, 1\}$ with

$k \in \mathbb{N} - \{0\}$ or $Q = \{r \in \mathbb{R} : 0 \leq r \leq 1\}$. Let the connectives be defined as at the end of Chapter 1.5. Now a countable set S of formulas is satisfiable iff S is finitely satisfiable.

Proof. Let $\alpha_1, \alpha_2, \ldots$ be an enumeration of the formulas in S and let $A_1, A_2, \ldots$ be an enumeration of all logical atoms. Assume that S is finitely satisfiable. Then there is for each n a truth-assignment ν_n with $\overline{\nu}_n(\alpha_m) = 1$ for $m = 1, 2, \ldots, n$.

One equips the space of all truth-assignments with the following metric: If $\mu = \mu'$ then $d(\mu, \mu') = 0$ else $d(\mu, \mu') = \min\{2^{-k} : k \in \mathbb{N}$ for all $\ell \leq k$ with $\ell \neq 0$, $\mu(A_\ell)$ and $\mu'(A_\ell)$ differ at most by $1/k\}$. The space of all truth-assignments with this metric is a complete compact metric space (and this property stems also from the fact that Q is compact). Thus it has the property that every infinite sequence of truth-assignments has a convergent subsequence. In particular one can select from the sequence of ν_n a subsequence of μ_m such that each μ_m equals to some ν_n with $n \geq m$ and the μ_m converge with respect to the metric d; they therefore in particular converge pointwise to a limit μ. As every function B_{α_k} is continuous and is based on finitely many atoms, it follows also that the limit μ of all μ_m satisfies that $\overline{\mu}(\alpha_k) = \lim_m \overline{\mu}_m(\alpha_k)$. As $\overline{\mu}_m(\alpha_k) = 1$ for all $m \geq k$, it follows that $\overline{\mu}(\alpha_k) = 1$. Hence μ is a truth-assignment which makes all members of S true and μ witnesses that S is satisfiable. $\square$

Notions of Effectiveness. One can ask in general when it is possible to compute something. For the world of natural numbers, there are several equivalent approaches.

1. There is a program in a usual programming language, say Javascript, which computes the function, where one permits the following constructs:

    - Input parameters in function declaration;
    - Some computations involving adding, subtracting and comparing variables and usage of new variables;
    - If-then-else commands (conditional branching) with conditions which are Boolean combinations of comaprisons of variables with each other or constants;
    - Loops which are either while-loops or for-loops with conditions as above;
    - Nested versions of the before mentioned constructs;
    - Calls of subfunctions;

- A natural number as output which is returned by a special command at the end of the loops.

The main data-type for this is the notion of the natural number; the program execution time is not bounded by any function and the output of a function is undefined when the program runs forever.

2. One defines the function from some basic functions by recursion in one variables where $+$ and $-$ and comparison-functions (output 1 or 0 depending on whether a comparision is true or false) are defined; for example, multiplication is defined recursively by $x \cdot 0 = 0$ and $x \cdot (y+1) = (x \cdot y) + x$. Furthermore, if f is a recursive function (say, with inputs x, y, z), then one can define a new function which maps x, y to the least z such that $f(x, y, z) = 0$; note that the function can have undefined values when such a z cannot be found.

3. A function f whose graph is a Diophantine set; that is, there is a polynomial p with coefficients from $\mathbb{Z}$ such that $f(x) = y$ iff there are $z_1, \ldots, z_k$ (for some constant k) with $f(x, y, z_1, \ldots, z_k) = 0$. While the coefficients of the polynomial are integers which can be negative, the values for $x, y, z_1, \ldots, z_k$ are all from $\mathbb{N}$.

The third definition is much easier to understand than the first two, as it does not rely on a programming language and also not on the choice of recursion. However, the third definition is much more difficult to handle; it took until 1970 when Matiyasevich showed the unsolvatibility of Hilbert's Tenth problem that mathematicians realised that all partial recursive functions are actually giuven by a Diophantine set. The notions of algorithms and definitions by recursion on the natural number were, however, already established before 1940.

Decidable and Recursively Enumerable Sets. The notion of an algorithm (version 1 in the above) permits to bring the concept of recursive (or effective or computable) functions from the domain of numbers into the domain of words, strings and formulas, as one can employ the usual string-operations from programming languages during the process of computation. Already in the early days of the research on effectivity, one noted that there are two versions of sets which are different, but come up in many contexts all the time:

1. Decidable sets L (also called recursive set L): This is a set for which there is a recursive function f such that, for all inputs x, $f(x) = 1$ when $x \in L$ and

$f(x) = 0$ when $x \notin L$. That is, some algorithm can determine which possible intputs are in the set L and which are not.

2. Recursively enumerable sets L (also called effectively enumerable sets L): There is an algorithm (that is, recursive function) which enumerates the members of L; for avoiding problems, one also defines that the empty set $\emptyset$ is recursively enumerable.

On the domain $\mathbb{N}$, Matiyasevich showed that the notion of a Diophantine set is the same as a recursively enumerable set; here a set A is Diophantine iff there is a polynomial p with coefficients from $\mathbb{Z}$ and inputs $x, y_1, \ldots, y_n$ from $\mathbb{N}$ such that $x \in A$ if and only if there are $y_1, \ldots, y_n \in \mathbb{N}$ with $p(x, y_1, \ldots, y_n) = 0$. This would permit to give the defintion of a recursively enumerable set again without any mention of machines, allowed programming commands in algorithms, run-time and other such ingredients. Further, equivalent definitions are that A is the domain of a partial-recursive function or the range of a partial-recursive function. It is easy to see that every decidable set is recursively enumerable.

Church's Thesis. All "reasonable formalisations" of the notion of an algorithm are equivalent; for all of these formalisations, the resulting notions of "recursively enumerable sets" coincide with each other; similarly the notions of "decidable sets" coincide with each other. This in particular means that one can employ larger sets of permitted commands in algorithms than the above given ones. However, when considering proofs over properties of partial-recursive functions, one keeps the amount of permitted operations limited.

Recursively enumerable sets which are not recursive. Since the early days of recursion theory, one knows that there are sets which are recursively enumerable but not recursive. The most famous example of this type is Turing's halting problem: It is the set of all pairs of computer programs e and inputs x such that a machine executing program e on input x will eventually halt and produce some output. A related undecidable but recursively enumerable set is the diagonal halting problem where one feeds into the computer program not an arbitrary input x but the own code e as input. Turing gave these examples in the year 1936.

Theorem 17B. There is an algorithm which can check whether an expression α is a well-formed formula; that is, the set of all well-formed formulas is decidable.

Comment. Here might be a problem that there are infinitely many atom symbols, so that one cannot represent the above set with a finite alphabet as it would be needed to process a formula in a computer. A way out is that one does not write $A_1, A_2, A_3, \ldots$, but to write $A, A', A'', \ldots$ and the number of primes of the A allow to distinguish the atoms. This then allows to use a finite alphabet as required by most of the text-processing algorithms.

Theorem 17C. Given a finite set of formulas S and a formula α, one can decide whether $S \models \alpha$.

Proof. Note that each formula $\beta \in S$ as well as α has only finitely many atoms. Thus there are only finitely many atoms which occur in either β or α. Now one can make a truth-table involving all these atoms and then check whether all rows which make all formulas in S true, also make α true. If so then $S \models \alpha$ else one has a counter-example row which tells how to choose the truth-values of the relevant atoms in order to make all members of S true while α becomes false. □

Corollary 17D. If S is finite then the set $\{\alpha : \alpha \text{ is a wff and } S \models \alpha\}$ is decidable.

Theorem 17E. A set is recursively enumerable iff there is an algorithm which, for all $x \in A$, outputs "yes"; however, for $x \notin A$, the function $f(x)$ might either never output something or output "no".

Proof. Assume that A is the range of a recursive function f. Now one can make the following algorithm:

Input x;
Let $y = 0$;
While $f(y) \neq x$ Do Begin $y = y + 1$ End;
Output "yes".

This algorithm outputs "yes" iff it finds an y with $f(y) = x$; note that it can evaluate $f(y)$ for each $y \in \mathbb{N}$ in finite time and therefore terminates eventually when there is such an y. However, if there is no y with $f(y) = x$, it will run forever. In the case that $A = \emptyset$, one just makes an algorithm which on all inputs outputs "no".

For the other way round, in the case that $A = \emptyset$, A is recursively enumerable by definition. Otherwise there is a fixed element $a \in A$. Now for each input x,

one makes a two-input function $f(x,t)$. This function simulates the algorithm on input x for t seconds. If the algorithm halts within t seconds with output "yes" then $f(x,t) = x$ else $f(x,t) = a$. It is easy to see that only members of A are in the range of f. Furthermore, if $x \in A$ then the algorithm needs some time t to say "yes" and therefore for all $s > t$, $f(x,s) = x$; hence the range of f is exactly the set A. □

Theorem 17F (Kleene's Theorem). A set A is decidable iff both the set A and its complement are recursively enumerable.

Proof. If A is decidable, there is an algorithm which on input x does the following: If $x \in A$ then the algorithm eventually outputs "yes" else the algorithm eventually outputs "no". This algorithm directly shows by Theorem 17E that A is recursively enumerable; when interchanging "yes" and "no" at the answer, one also gets an algorithm which witnesses that the complement of A is recursively enumerable.

For the converse direction, note that when $A = \emptyset$ or $A = \mathbb{N}$ there are trivially algorithms which show that A is decidable. Thus assume now that A is neither $\emptyset$ not $\mathbb{N}$ and A is the range of the recursive function f and its complement is the range of the recursive function g. Without loss of generality, the domains of f and g are both $\mathbb{N}$. Now one makes the following algorithm:

Input x;
Let $y = 0$;
While ($f(y) \neq x$ and $g(y) \neq x$) Do Begin $y = y + 1$ End;
If $f(y) = x$ then output "yes" else output "no".

Since $x \in range(f) \cup range(g)$, the algorithm will eventually find a y such that the condition of the while-loop is not satisfied. Once this y is reached, the if-then-else-statement will provide the correct output: If y is in the range of f then the answer is "yes" else y is in the range of g and the answer is "no". □

Generalisations of Corollary 17D. One might ask what happens if one does not restrict S to finite sets. As there are $\aleph_0$ atoms, one can select $2^{\aleph_0}$ many sets of atoms and negated atoms; however, there are only $\aleph_0$ many algorithms, as each algorithm can be written down in as a finite text over a fixed alphabet. Thus most of these sets are not decidable, as there are less algorithms than sets. Thus it is reasonable to require that one looks only at sets S which are themselves enumerated by an algorithm. But even then one might have a problem. Consider Turing's diagonal halting problem K:

A number e is in K iff the e-th computer program with input e halts and produces some output. Now one let $S = \{A_e : e \in K\}$; the set S is also recursively enumerable. It is easy to see that $S \models A_e$ iff A_e is enumerated into S iff $e \in K$. Thus the set of all formulas α such that $S \models \alpha$ is not decidable. However, one has the following result.

Theorem 17G. If S is a recursively enumerable set of formulas then $\{\alpha : \alpha$ is a wff and $S \models \alpha\}$ is recursively enumerable.

Proof. Let $\beta_1, \beta_2, \ldots$ be a recursive enumeration of all formulas and $\alpha_1, \alpha_2, \ldots$ be a recursive enumeration of the formulas in S. Now one considers the following function:

$$f(n,m) = \begin{cases} \beta_{m+1} & \text{if } \{\alpha_1, \ldots, \alpha_{n+1}\} \models \beta_{m+1}; \\ \alpha_1 & \text{if } \{\alpha_1, \ldots, \alpha_{n+1}\} \not\models \beta_{m+1}. \end{cases}$$

This function f is recursive, as there is a decision procedure which checks whether a finite set of formulas implies another one by Theorem 17C. Furthermore, by Corollary 17A, if $S \models \beta_{m+1}$ then there is a finite set $\{\alpha_1, \ldots, \alpha_{n+1}\} \subseteq S$ such that $S \models \beta_{m+1}$ and thus $f(n,m) = \beta_{m+1}$ for this formula β_{m+1}. Thus the range of f are exactly all the formulas which are tautologically implied by S. $\square$

Theorem 17H. If S is a recursively enumerable set of formulas then there is a further decidable set T of formulas with

$$\forall \text{ wff } \alpha\, [S \models \alpha \text{ if and only if } T \models \alpha].$$

Proof. If S is finite then S is already decidable and nothing needs to be done. So assume that S is infinite and S is the range of a recursive function f, so $S = \{f(0), f(1), \ldots\}$ where f maps the natural numbers to wffs. Now let $g(n)$ be the formula $f(n) \wedge (A_{n+1} \leftrightarrow A_{n+1})$. It is easy to see that $\{g(n)\} \models f(n)$ for all n, as $g(n)$ is just the conjunction of $f(n)$ and a tautology. Furthermore, $\{f(n)\} \models f(n)$ and $\{f(n)\} \models (A_{n+1} \leftrightarrow A_{n+1})$; thus $\{f(n)\} \models g(n)$. So one can conclude that for all n and all $m \geq n$, the following conditions hold:

$$\begin{aligned} \{f(0), f(1), \ldots, f(m)\} &\models g(n); \\ \{g(0), g(1), \ldots, g(m)\} &\models f(n). \end{aligned}$$

Thus, if $S \models \alpha$ then there is by the compactness theorem some m such that the following statements are true:

$$\begin{aligned} \{f(0), f(1), \ldots, f(m)\} &\models \alpha; \\ \{g(0), g(1), \ldots, g(m)\} &\models f(n) \text{ for all } n \leq m; \\ \{g(0), g(1), \ldots, g(m)\} &\models \alpha; \\ T &\models \alpha. \end{aligned}$$

Similarly, one can show that under the assumption $T \models \alpha$ there is an m such that the following statements hold:

$$\begin{aligned} \{g(0), g(1), \ldots, g(m)\} &\models \alpha; \\ \{f(0), f(1), \ldots, f(m)\} &\models g(n) \text{ for all } n \leq m; \\ \{f(0), f(1), \ldots, f(m)\} &\models \alpha; \\ S &\models \alpha. \end{aligned}$$

Now it remains to show that T is decidable. Indeed, T has the following decision procedure.

1. On input α, determine the largest n such that A_n appears in α; $n = 0$ is if there is no atom in α;
2. For $m = 0, 1, \ldots, n$ Do Begin check whether $g(m) = \alpha$ End;
3. If an $m \leq n$ was found with $g(m) = \alpha$ then output "yes" else output "no".

The first statement can be done effectively, as one has only to go through the formula and literally just inspect all the atoms which occur there. The second statement can be done effectively, as g is a recursive function which is defined on all natural numbers and computed in finite time by some computer program and comparing formulas (strings) is a standard task which can be done by computers. Furthermore, there are only $n+1$ of these comparisons, so the loop terminates. The third statement can be done effectively, as one just has to note down the results of the comparisons in Step 2 and output "yes" is any of these results said "equal" and output "no" if all of them said "not equal". Thus T is a decidable set. $\square$

Closure Properties. The following statements are true for recursively enumerable and decidable subsets of $\mathbb{N}$:

(a) Every infinite recursively enumerable set X has an infinite recursive subset Y;

(b) If X and Y are both recursively enumerable, so are $X \cup Y$ and $X \cap Y$;

(c) If X and Y are decidable, so are $X \cup Y$, $X \cap Y$ and $\mathbb{N} - X$.

Proof of (a). Given X, one can make a recursive function doing the following: $f(0)$ is the minimum of X; $f(n+1)$ is the first element y of X found by effective search which satisfies $f(n) < y$; note that this search always terminates eventually, as X is infinite. Thus all values of f can be computed in finite time and now let Y be the range of f. Clearly Y is recursively enumerable. One can, however, even show that Y is decidable. For this, note that $f(y) \geq y$ for all y as one can show that f is strictly monotonically increasing. Thus one has for all y that

$$y \in Y \Leftrightarrow y \in \{f(0), f(1), \ldots, f(y)\}$$

and this completes the proof of (a).

Proof of (b). Let X and Y be recursively enumerable sets. If one of them is empty, their union is the other set and their intersection is $\emptyset$. If both are non-empty then they are the ranges of recursive functions f_X and f_Y. Now $X \cup Y$ is the range of the function f defined by $f(2n) = f_X(n)$ and $f(2n+1) = f_Y(n)$. Thus $X \cup Y$ is also recursively enumerable. Furthermore, if $X \cap Y = \emptyset$ then $X \cap Y$ is recursively enumerable by definition; if $X \cap Y \neq \emptyset$ then let a be one of the elements in the intersection and now define

$$g(n,m) = \begin{cases} a & \text{if } f_X(n) \neq f_Y(m); \\ f_X(n) & \text{if } f_X(n) = f_Y(m). \end{cases}$$

Now it is easy to see that g is a recursive function with range $X \cap Y$ and so $X \cap Y$ is recursively enumerable.

Proof of (c). Given decision procedures for X and Y, one can build the following decision procedures for $X \cup Y$, $X \cap Y$ and $\mathbb{N} - X$: Assume that $X(n) = 1$ when $n \in X$ and $X(n) = 0$ whenb $n \notin X$, similarly for Y. That is, one identifies the sets with their characteristic function. Now one can just define the decision procedures for intersection, union and complement by the below list using those for X and Y:

1. $(X \cap Y)(n) = \min\{X(n), Y(n)\}$;
2. $(X \cup Y)(n) = \max\{X(n), Y(n)\}$;
3. $(\mathbb{N} - X)(n) = 1 - X(n)$.

This completes the proof. □

Alternative proof for Theorem 17H. Assume that S is recursively enumerable, that is, $S = range(f)$ for a recursive function from $\mathbb{N}$ into the set of all wff. Then one can also consider $T = range(g)$ with $g(0) = f(0)$ and $g(n+1) = (g(n) \wedge f(n+1))$. As g is made by recursion on f, g is also recursive. Furthermore, for all m, n with $m \leq n$, $\{f(0), f(1), \ldots, f(n)\} \models g(n)$, $\{g(n)\} \models f(m)$ and $\{g(n)\} \models g(m)$. Thus it holds that if $S \models \alpha$ then there is a finite subset $\{f(0), f(1), \ldots, f(n)\}$ which tautologically implies α and furthermore $\{g(m)\} \models \alpha$ for all $m \geq n$. Furthermore, T is also recursively enumerable and thus T has an infinite decidable subset U. Now, for every wff α, $U \models \alpha$ iff $S \models \alpha$. This then completes the proof of Theorem 17H using the above closure properties.

Chapter 2 – First-Order Logic

While sentential logic only deals with connections between sets of formulas using atoms as truth-values, first-order and second-order logic try to replace these atoms by statements over some mathematical structure. Such statements have in particular some basic ingredients:

1. A base set X in which the structure lives;
2. Basic operations on the set X, say some functions;
3. Predicates on X or $X \times X$ or $X \times X \times X$ and so on which can cut-out subsets of X or define important relations including equality;
4. Constants which denote elements of X of special importance;
5. Quantifiers, namely existential quantifiers ($\exists$) and universal quantifiers ($\forall$) which allow to make statements about mathematical laws valid on X using the aforementioned predicates and functions.

An example of first-order and second-order statements are the following rules which hold to define within a structure $(X, \mathbb{N}, Succ, 0, 1)$ the set $\mathbb{N}$ of natural numbers:

1. $0 \in \mathbb{N}$ (zero is a natural number);
2. $\forall x \in X\,[x = x]$ (for all objecs x, x is equal to itself);
3. $\forall x, y \in X\,[x = y \to y = x]$ (for all objects x, y, if x is equal to y then y is equal to x; that is, the equality predicate is commutative);
4. $\forall x, y, z \in X\,[x = y \wedge y = z \to x = z]$ (transitivity of the equality);
5. $\forall x, y \in X\,[(x \in \mathbb{N}) \wedge (x = y) \to (y \in \mathbb{N})]$ (if two objects are equal and one of them is a natural number, then actually both of them are a natural number);
6. $\forall x \in \mathbb{N}\,[Succ(x) \in \mathbb{N}]$ (the successor-function maps natural numbers to natural numbers);
7. $\forall x, y \in \mathbb{N}\,[x = y \leftrightarrow Succ(x) = Succ(y)]$ (two natural numbers are the same iff their successors are the same);

8. $\forall x \in \mathbb{N}\,[0 \neq Succ(x)]$ (zero is not the successor of any natural number; in particular, -1 is not a natural number);

9. $\forall K \subseteq X\,[0 \in K \wedge \forall x \in K\,[Succ(x) \in K] \rightarrow \forall x \in \mathbb{N}\,[x \in K]]$ (if a set contains 0 and contains for every x also its successor, then the set contains all natural numbers).

These are the nine axioms listed as Peano's axioms for natural numbers on Wikipedia. Usually, one assumes that any logic satisfies this axioms and that $x = y$ holds iff x and y denote the same element. All axioms except for the ninth are only quantifying over members of the structure; the ninth axiom, however, quantifies over sets. First-order logic does not allow it; only second-order logic allows to quantify over sets or functions. One has therefore studied in logic, whether one can describe the natural numbers uniquely with axioms from first-order logic and the result is that this is impossible. There are many structures which are very similar to the natural numbers and satisfy all the first-order formulas which are true for the natural numbers, but which are not isomoprhic to the natural numbers and the successor-relation. Quantifyers had already been used informally; the following examples, however, should make their meaning more clear.

Examples of quantified formulas. Let X be a base set and $f : X \rightarrow X$ be any function. The following rules about f are now formulated using first-order logic and equality.

1. $\forall y \in X \exists z \in X\,[f(z) = y]$. For every y in X there is a z in X such that $f(z)$ takes the value y; that is, f is an onto function from X to itself.

2. $\exists y, z \in X\,[y \neq z \wedge f(y) = f(z)]$. There are y, z in X such that y and z are not equal, but their images under f is; that is, f is not injective.

3. $\forall y, z \in X\,[f(y) = f(z) \leftrightarrow y = z]$. For all y, z in X, $f(y) = f(z)$ if and only if $y = z$; that is, f is injective.

4. $\forall y \in X\,\exists z \in X\,[(z = y \vee z = f(y)) \wedge f(z) = y]$. For all y in X there is a z in X such that z is either y or $f(y)$ and $f(z)$ equals to y; that is, every value is mapped to itself by either once or twice applying f.

The reference to the base set X in the quantification can be omitted in the case that the range of the quantifyer is clear. The range of the quantifier is only included in the case that there are several choices and in the case that a base set is fixed, the range of the quantifier is always the base set unless something else is specified. So one could write the first two examples also as follows:

1. $\forall y \exists z \, [f(z) = y]$;
2. $\exists y, z \, [y \neq z \wedge f(y) = f(z)]$.

When using $+$, $\cdot$ or other binary operations of numbers, these are in an informal setting just written as usually in mathematics, together with the rules that $\cdot$ binds more than $+$ and $-$; however, when doing it formally, one has to replace $x + y$ by a function $add(x, y)$ and $x \cdot y$ by a function $mult(x, y)$ in order to avoid too much complications when making proves over formulas. Furthermore, the number of connectives can be cut down to a generating set of connectives; most convenient is for this the set $\{\neg, \rightarrow\}$ of the negation and the implication; one could also use $\{\neg, \wedge, \vee\}$. Furthermore, one replaces each formula of the form $\exists x \, [\alpha]$ by $\neg \forall x \, [\neg \alpha]$ and therefore one does not need both quantifiers, but only one; however, in an informal setting, one can still use both quantifiers.

Chapter 2.1 – First-Order Languages

Symbols in language.

A. Logical Symbols. These denote the logical language and also variables ranging over the structure.

A.0. Brackets: (and); after quantification [and] in an informal setting.

A.1. Sentential connective symbols: $\rightarrow$, $\neg$; the others only in an informal setting.

A.2. Variables $v_0, v_1, \ldots$ which range over members of the underlying structure.

A.3. Equality symbol $=$ which is used in most but not all cases.

B. Parameters. These denote item specific to the structure used and the quantifiers.

B.0. Quantifier symbol $\forall$; the symbol $\exists$ occurs only in informal settings.

B.1. Predicate symbols for handling subsets of the domain and also for handling relations and each predicate symbol comes with an arity of the inputs and these are inside brackets and separted by commas.

B.2. Constant symbols (like $0, 1$ in the natural numbers or fields).

B.3. Function symbols (in a formal setting, operators like $+$ and $\cdot$ and $-$ are replaced by functions) and each function symbol comes with an arity, that is, the number of inputs which are in brackets and separated by commas.

The equality symbol can be present, but one can also consider logics without equality, though those are only a minor topic in this course. In the following some examples of languages for which one specifies what symbols are in.

Example: Language of Two Predicates. No equality. Symbols A_3, A_4 for a three-place and a four-place predicate: $A_3(u, v, w)$, $A_4(u, v, w, x)$. Constant symbols: $a_1, a_2, a_3, \ldots$ but no function symbol. As a sample formula, one gives one which states that the three-way formula always holds when the four-way predicate holds with an additional existential quantification over the fourth input:

$$\forall v_1, v_2, v_3 \, \exists v_4 \, [A_4(v_1, v_2, v_3, v_4) \rightarrow A_3(v_1, v_2, v_3)].$$

This formula can then be translated into the usual normalform as follows:

$$(\forall v_1\,(\forall v_2\,(\forall v_3\,(\neg(\forall v_4\,(\neg(A_4(v_1,v_2,v_3,v_4)\to A_3(v_1,v_2,v_3))))))).$$

One can also consider languages with more than two predicates; in principle, even languages with an infinite number of predicates of infinitely many distinct arities are possible. However, there should be an algorithm which says for which arity how many predicates exist and how these are called.

Example: Language of Set Theory. Has equality. One predicate $\in$ to denote membership in a set. One constant symbol $\emptyset$ to denote the emptyset. Here some sample statements.

- There is no set containing all sets:

  $$\forall v_1 \exists v_2\,[v_2 \notin v_1].$$

  More formally, one has to eliminate the existential quantifier and to replace $v_2 \notin v_1$ by $\neg v_2 \in v_1$ which, after eliminatation of double negation, gives the following statement:

  $$(\forall v_1\,[\neg(\forall v_2\,[v_2 \in v_1])]).$$

  A last step would be to replace the square brackets by normal brackets.

- For any two sets there is a set which contains exactly these two sets as an element:

  $$\forall v_1\,\forall v_2\,\exists v_3\,\forall v_4\,[v_4 \in v_3 \leftrightarrow (v_4 = v_1 \vee v_4 = v_2)]$$

  which one first transforms into

  $$\forall v_1\,\forall v_2\,\neg\forall v_3\,\neg\forall v_4\,[v_1 \in v_3 \wedge v_2 \in v_3 \wedge (v_4 \in v_3 \to (v_4 = v_1 \vee v_4 = v_2))].$$

  For atoms, one has the rules that a sequence $A_1 \wedge A_2 \wedge A_3$ gives $A_1 \to A_2 \to \neg A_3$ and furthermore $\neg(A_4 \to (A_5 \vee A_6))$ is $\neg(A_5) \wedge \neg(A_6) \wedge A_4$. Combining these rules, one can transform the above formula into the following quantified sequence of implications:

  $$\forall v_1\,\forall v_2\,\neg\forall v_3\,\neg\forall v_4\,[v_1 \in v_3 \to v_2 \in v_3 \to v_4 \neq v_1 \to v_4 \in v_2 \to v_4 \notin v_3].$$

Now making all brackets and replacing all $\neq$ and $\notin$ gives the formula

$$(\forall v_1\,(\forall v_2\,(\neg(\forall v_3\,(\neg(\forall v_4\,[\neg((v_1 \in v_3) \to ((v_2 \in v_3) \to ((\neg(v_4 = v_2)) \to ((\neg(v_4 = v_1)) \to (\neg(v_4 \in v_3))))))]))))))).$$

Example: The language of Abelian group theory. The language consists of some constants for special members of the group, in particular 0 for the neutral element, plus a symbol + for addition. The language also uses the equality. In the following, one gives the axioms for a commutative group $(A, +, 0)$ where 0 is the only constant used:

$\forall v_1, v_2, v_3\,[v_1 + (v_2 + v_3) = (v_1 + v_2) + v_3]$;
$\forall v_1, v_2\,[v_1 + v_2 = v_2 + v_1]$;
$\forall v_1\,[v_1 + 0 = v_1]$;
$\forall v_1\,\exists v_2\,[v_1 + v_2 = 0]$.

One has then to transform these formulas into the required form by using a function *add* for the addition in place of the operator +; that is, the official language would be the usual symbols plus the constant 0 and the function *add* and equality. Furthermore, one uses $\neg\forall\neg$ for $\exists$. For example, the last formula becomes

$$(\forall v_1\,(\neg(\forall v_2(\neg(add(v_1, v_2) = 0))))).$$

The other formulas are translated similarly into this normal form.

A systematic approach to well-founded formulas. In first-order logic, one first has to deal with the terms and atomic formulas on the level of the structure and then, afterwards, to form general wffs from it. In general, any string over the alphabet is called an expression and those, but only those which are formed by applying finitely often constructorfunctions from base formulas are interesting and wffs.

Terms. A constant or a variable of the first-order language are a term. Furthermore, for every n-ary function symbol f and n terms $t_1, t_2, \ldots, t_n$, the expression $f(t_1, t_2, \ldots, t_n)$ is also a term, that is, given the n terms, one obtains a new term by separating out the terms with commas, adding a closing bracket at the end and the symbol f plus an opening bracket at the front. For example, if *add* is the two-input addition function on the natural numbers and $0, 1$ are the allowed constants, then

the following expressions are terms: 0, 1, $add(0,1)$, $add(1,1)$, $add(1,add(1,1))$ and $add(add(1,1),1)$ which represent the numbers 0, 1, 1, 2, 3 and 3, respectively. Rules to evaluate and compare the terms will come later. If there is a further function $mult$, then also $mult(add(1,1),add(1,1))$ would be a term which would provide the value 4. Here "add" and "mult" and "succ" are supposed to stand for symbols of length 1 denoting the corresponding operations, they are spelled out for better readability.

Atomic Formulas. Formulas which are either of the form $P(t_1,t_2,\ldots,t_n)$ for an n-ary predicate and n terms or of the form $t_1=t_2$ or of the form q for a truth-value q are called atomic formulas (truth-values could be considered as nullary predicates). When dealing with set theory, there is a predicate denoting the element-relation; however, one writes usually expressions like $t_1 \in t_2$ and is not using a predicate like $in(t_1,t_2)$, although the latter would be the standard form.

Well-Formed Formulas. These are formed from atomic formulas by putting them together with logical connectives (formally one uses only $\neg$ and $\rightarrow$, as the other can be expressed using these) or with quantifiers (formally, one uses only $\forall$, as a formula of the form $\exists v_k\,[P(v_k)]$ is equivalent to $\neg(\forall v_k\,[\neg P(v_k)])$ and similarly for more complicated formulas in place of $P(v_k)$). Formally, well-formed formulas have always outmost brackets which are kept when assembling larger formulas from parts.

Free Occurrences of Variables. All occurrences of variables in atomic formulas are free. An occurrence of a variable v_i within the range of a quantifier $\forall v_i\,[\alpha]$ is bound. Furthermore, if v_i occurs free within α and, respectively, β, then these occurrences are also free in the formulas $\neg\alpha$ and $\alpha\rightarrow\beta$. For example, the occurrences of v_1, the last occurrence of v_2 and the first two occurrences of v_3 are free in the formula

$$((\forall v_2\,[(v_1=v_3)\rightarrow(v_2=v_3)])\rightarrow(\neg\forall v_3\,[\neg P(v_1,v_2,v_3)]))$$

the variables v_2, v_3 are bound within the range $[\ldots]$ after the corresponding quantifiers. The free variables inside a formula can be defined using the following recursively defined function h, where v_i stands for a variable, c_j for a constant, $t_1,t_2,\ldots,t_n$ for terms, f for an n-ary function and P for an n-ary predicate and α,β for wffs.

1. $h(v_i)=\{v_i\}$;
2. $h(c_j)=\emptyset$;

3. $h(f(t_1, t_2, \ldots, t_n)) = h(t_1) \cup h(t_2) \cup \ldots \cup h(t_n)$;

4. $h(t_1 = t_2) = h(t_1) \cup h(t_2)$;

5. $h(P(t_1, t_2, \ldots, t_n)) = h(t_1) \cup h(t_2) \cup \ldots \cup h(t_n)$;

6. $h(\neg\alpha) = h(\alpha)$;

7. $h(\alpha \to \beta) = h(\alpha) \cup h(\beta)$;

8. $h(\forall v_k\,[\alpha]) = h(\alpha) - \{v_k\}$.

Note that the recursive applications of the first four items gives that if α is an atomic formula then $h(\alpha)$ is the set of all variables which occur in α. The last three items define h inductively on wffs from the definition of h on atomic formulas.

Every wff α which satisfies $h(\alpha) = \emptyset$ is called a sentence. When permitting all connectives and quantifiers, the following formulas are sentences:

1. $\forall v_1 \exists v_2\,[v_1 = v_2]$;

2. $\forall v_1 \exists v_2\,[\neg(v_1 = v_2)]$;

3. $\exists v_1 \exists v_2 \forall v_3\,[v_3 = v_2 \vee v_3 = v_1]$;

4. $\forall v_1\,[\neg(v_1 = v_1)]$.

Among these sentences, the first is a tautology and the last is an antitautology. Furthermore, the second sentence holds in structures with at least two elements and the third holds in structures with at most two elements.

Additional Conventions. As done in the text before, for general writing of formulas, in the case of several quantifiers having the same range like $\forall v_1\,[\exists v_2\,[v_1 = v_2]]$, one writes only one set of squarebrackets, namely the inner one as in $\forall v_1\,\exists v_2\,[v_1 = v_2]$. Furthermore, $t_1 \neq t_2$ refers to $\neg(t_1 = t_2)$ and in formulas, the outmost bracket can be omitted. One can also write $v_1 + v_2$ in place of $add(v_1, v_2)$ and similarly for further functions with two inputs representing group and monoid operations. When there are additive and multiplicative operations, the multiplicative operations bind more, so $v_1 + v_2 \cdot v_3$ is $v_1 + (v_2 \cdot v_3)$. In doubt, brackets are placed for making the meaning of a formula clear.

Chapter 2.2 – Truth and Models

Structures and Models. A "model" and a "structure" are almost the same. The main difference is that structure only refers to itself while model is also referring to a set of formulas and one says that $\mathfrak{A}$ is a model of a set S of formulas, if $\mathfrak{A}$ makes all formulas in S true.

A structure $\mathfrak{A}$ consists of a nonempty domain A which assigns to every constant, function and predicate a value from A (constants) or from functions from A^n to A (n-ary function symbols) or from functions from A^n to the set Q of truth-values (n-ary predicates); for most of this lecture, $Q = \{0, 1\}$. For constant-symbols c, function-symbols f and predicate-symbols P, $c^{\mathfrak{A}}$, $f^{\mathfrak{A}}$ and $P^{\mathfrak{A}}$ denote the objects associated with c, f and P in the structure $\mathfrak{A}$.

Examples. When one considers the structure of natural numbers with the function add and constants 0 and 1. Then the domain A of $\mathfrak{A}$ is $\mathbb{N}$ and the function $add^{\mathfrak{A}}$ maps pairs of natural numbers x, y to $x + y$. The constants $0^{\mathfrak{A}}$ and $1^{\mathfrak{A}}$ take as values just the smallest and second-smallest natural numbers which usually are called "zero" and "one". The term $add(1, 1)^{\mathfrak{A}}$ has the value 2. The following formula is true in the natural numbers:

$$\exists x \, \forall y, z \, [add(y, z) = x \to (y = x \wedge z = x)].$$

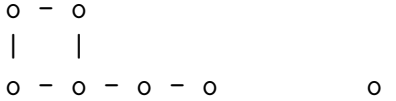
The reason is that 0 can only be the sum of 0 and 0. So one says that this sentence is true in $\mathfrak{A}$.

On the other hand, for the structure $\mathfrak{B}$, the function $add^{\mathfrak{B}}$ is the addition on integers and there, every number x is the sum of -1 and $x + 1$ or, equivalently, of -2 and $x + 2$. Thus above sentence is wrong. Indeed,

$$\forall x \, \exists y, z \, [add(y, z) = x \wedge y \neq x \wedge z \neq x]$$

it true in the structure $\mathfrak{B}$.

Example. Consider a graph given by its edge set E. For example, as given in the following diagramme.



The edge relation E is not by default undirected, instead any binary relation would be a directed edge relation. So one would have to require that an undirected graph satisfies the following axiom:

$$\forall x \, \forall y \, [E(x, y) \to E(y, x)].$$

Above example satisfies that there are nodes of degree 0, degree 1, degree 2 and degree 3, where the degree of a node is the number of neighbours. So having a node of at least degree 3 could be expressed as follows:

$$\exists x \, \exists u \, \exists v \, \exists w \, [u \neq v \wedge u \neq w \wedge v \neq w \wedge E(x, u) \wedge E(x, v) \wedge E(x, w)].$$

The sentence that every node has at least degree 1 is as follows:

$$\forall x \, \exists y \, [E(x, y)].$$

The graph from above diagramme does not satisfies this sentence, but the two previous sentences of this example (undirectedness and existence of node with three neighbours).

Defining values of terms. A structure $\mathfrak{A}$ and a default assignment s which assigns to every variable a value from the domain A of A_k allow to assign to every term t a value $\overline{s}(t)$. This is done by induction:

1. For every variable $x \in \{v_1, v_2, \ldots\}$, $\overline{s}(x) = s(x)$.
2. For every constant c, $\overline{s}(c) = c^{\mathfrak{A}}$.
3. For every n-ary function symbol f, $\overline{s}(f(t_1, \ldots, t_n) = f^{\mathfrak{A}}(\overline{s}(t_1), \ldots, \overline{s}(t_n))$.

This now allows to define the truth-values of atomic formulas; if such a formula ψ is true, one writes $\mathfrak{A}, s \models \psi$.

Now one says that $\mathfrak{A}, s \models t_1 = t_2$ iff $\overline{s}(t_1) = \overline{s}(t_2)$. For an n-ary predicate P, one says that $\mathfrak{A}, s \models P(t_1, \ldots, t_n)$ iff $P^{\mathfrak{A}}(\overline{s}(t_1), \ldots, \overline{s}(t_n))$ is true. This defines the truth-value of atomic formulas.

If logical connectives, the same rules as in the case of sentential logic apply. These are the following:

$$\begin{aligned}
\mathfrak{A}, s \models (\neg\alpha) &\Leftrightarrow \mathfrak{A}, s \not\models \alpha; \\
\mathfrak{A}, s \models (\alpha \to \beta) &\Leftrightarrow \mathfrak{A}, s \not\models \alpha \text{ or } \mathfrak{A}, s \models \beta; \\
\mathfrak{A}, s \models (\forall x \, [\alpha]) &\Leftrightarrow \text{For all } d \in A, \, \mathfrak{A}, s(x|d) \models \alpha.
\end{aligned}$$

Here $s(x|d)$ is the modified default-assignment to variables where x gets the default d and all other variables y keep their default $s(y)$. Note that the other logical connectives and $\exists$ are expressed in terms of $\forall, \neg, \rightarrow$ and therefore it is sufficient to give the above definitions.

Example. Let $\mathfrak{A}$ be the structure $(\mathbb{N}, add, leq)$ of natural numbers with function add to add numbers and predicate leq to compare two terms with respect to whether the first term is less or equal the second term. Now one considers the following sentence:

$$(\forall x\,(\forall y\,(\neg\forall z\,(\neg leq(add(8, add(x, y)), z))))).$$

This formula would be, when written the usual way, just be

$$\forall x\,\forall y\,\exists z\,[8 + (x + y) \leq z]$$

and it is easy to see that this sentence is true. Now let s say that x has the default 1, y has the default 5 and z has the default 35. So $\overline{s}(add(x, y)) = 6$, $\overline{s}(8, add(x, y)) = 14$, $\overline{s}(z) = 35$ and $\mathfrak{A}, s \models leq(add(8, add(x, y)), z)$ or, when written the usual way, $\mathfrak{A}, s \models 8 + (x + y) \leq z$, as $14 \leq 35$. Furthermore,

$$\mathfrak{A}, s(x|a, y|b, z|c) \not\models \neg leq(add(8, add(x, y)), z)$$

and this holds in a more general way:

$$\mathfrak{A}, s(x|a, y|b, z|a + b + 9) \not\models \neg\neg leq(add(8, add(x, y)), z).$$

So there is, whatever one substitutes for x and y, the statement

$$\mathfrak{A}, s(x|a, y|b, z|c) \not\models \neg leq(add(8, add(x, y)), z).$$

is not true for all c and thus

$$\mathfrak{A}, s(x|a, y|b) \not\models \forall z\,(\neg leq(add(8, add(x, y)), z))$$

and

$$\mathfrak{A}, s(x|a, y|b) \models (\neg\forall z(\neg leq(add(8, add(x, y)), z))).$$

Now, when considering a quantifier over y one sees that

$$\mathfrak{A}, s(x|a, y|b) \models (\neg\forall z(\neg leq(add(8, add(x, y)), z)))$$

for all b and therefore

$$\mathfrak{A}, s(x|a) \models (\forall y\,(\neg\forall z(\neg leq(add(8, add(x, y)), z))))$$

and again, as this holds for all values $a \in A$, one can conclude that

$$\mathfrak{A}, s \models (\forall x\,(\forall y\,(\neg\forall z(\neg leq(add(8, add(x, y)), z))))).$$

Summarising informally,

$$\mathfrak{A}, s \models \forall x\,\forall y\,\exists z\,[8 + x + y \leq z]$$

can be made true for all values of x and y by having, in each case, z taking the value of the term $x + y + 9$.

If one has a formula with true defaults, say

$$\forall x\,\exists y\,[y = x + z]$$

then one has to observe that for all a it does not hold for all b that

$$\mathfrak{A}, s(x|a, y|b) \models (y \neq x + z)$$

and this is indeed the case by considering the case where $b = s(z) + a$. Therefore one can conclude that

$$\mathfrak{A}, s(x|a) \not\models (\forall y\,(y \neq x + z))$$

and

$$\mathfrak{A}, s(x|a) \models (\neg(\forall y\,(y \neq x + z)))$$

and, as this holds independently of the choice of a,

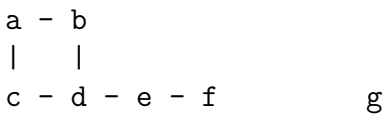
$$\mathfrak{A}, s \models (\forall x\,(\neg(\forall y\,(y \neq x + z))))$$

which is equivalent to

$$\mathfrak{A}, s \models (\forall x\,(\neg(\forall y\,(\neg(y = x + z)))))$$

which is then the same as the above informally written formula.

Example. Recall the graph given by its edge set E as in the following diagramme where now the vertices are named by letters.



Now one studies the following formulas with one free variable x:

$$\begin{aligned} \alpha &= \forall y \quad [\neg E(x,y)]; \\ \beta &= \exists y, z \quad [y \neq z \wedge E(x,y) \wedge E(x,z)]. \end{aligned}$$

Now $\mathfrak{A}, s \models \alpha$ iff $s(x) = g$ and $\mathfrak{A}, s \models \beta$ iff $s(x) \in \{a, b, c, d, e\}$. These example show that the influence of s on the truth of α and β only depends on the defaults of the free variable x and not on the defaults of variables which either do not occur in the formula or occur in the formula only in a bound form.

Theorem 22A. Given a structure $\mathfrak{A}$ and a formula α and two default assignments to the variables s_1 and s_2. If these agree on all variables which occur free in α then

$$\mathfrak{A}, s_1 \models \alpha \text{ if and only if } \mathfrak{A}, s_2 \models \alpha.$$

Proof. One proves this by induction over all well-formed formulas. If it is then true for all formulas, it is also true for α.

First consider an atomic formula $P(t_1, t_2, \ldots, t_n)$. For each term t_m, s_1 and s_2 take on the variables which occur free in the terms the same values. Thus $\overline{s}_1(t_m) = \overline{s}_2(t_m)$. It follows that $\mathfrak{A}, s_1 \models P(t_1, t_2, \ldots, t_n)$ if and only if $\mathfrak{A} \models P(\overline{s}_1(t_1), \overline{s}_1(t_2), \ldots, \overline{s}_1(t_n))$ if and only if $\mathfrak{A} \models P(\overline{s}_2(t_1), \overline{s}_2(t_2), \ldots, \overline{s}_2(t_n))$ if and only if $\mathfrak{A}, s_2 \models P(t_1, t_2, \ldots, t_n)$. Similarly for the case that the atomic formula is of the form $t_1 = t_2$.

If α and β satisfy the statement of the theorem, so do $\neg\alpha$ and $\alpha \to \beta$, as their truth-values are only Boolean combinations of those of α and β and the latter are the same for s_1 and s_2.

If $\alpha = \forall x\,[\beta]$, then one considers each variable assignment $s_1(x|a)$ and $s_2(x|a)$. In the case that s_1 and s_2 coincide on all free variables in α, then $s_1(x|a)$ and $s_2(x|a)$ coincide on all free variables in β. Thus, for all $a \in A$ (the domain of $\mathfrak{A}$), $\mathfrak{A}, s_1(x|a) \models \beta$ iff $\mathfrak{A}, s_2(x|a) \models \beta$ by induction hypothesis. Now, the following statements are equivalent:

1. $\mathfrak{A}, s_1 \models \alpha$;

2. for all $a \in A$ $[\mathfrak{A}, s_1(x|a) \models \beta]$;

3. for all $a \in A$ $[\mathfrak{A}, s_2(x|a) \models \beta]$;

4. $\mathfrak{A}, s_2(x|a) \models \alpha$.

Corollary 22B. If α is a sentence (no free variables) then either $\mathfrak{A}, s \models \alpha$ for all s or $\mathfrak{A}, s \models \alpha$ for no s.

Logical Implication. Let S be a set of wffs and α be a wff. Now one says that $S \models \alpha$ iff for every structure $\mathfrak{A}$ and every value-assignment s to the variables the following holds: If $\mathfrak{A}, s \models \beta$ for all $\beta \in S$ then $\mathfrak{A}, s \models \alpha$.

Here one says "S logically implies α". For simplification, one also writes $\mathfrak{A}, s \models S$ when one means that $\mathfrak{A}, s \models \beta$ for all $\beta \in S$. One says that two sets formulas α and β are logically equivalent iff $\{\alpha\} \models \beta$ and $\{\beta\} \models \alpha$. A formula α is valid iff $\emptyset \models \alpha$; so the valid formulas are those which are made true by all structures and defaults and they are the equivalent to the tautologies in sentential logic. A formula α is satisfiable iff there is a structure $\mathfrak{A}$ and a value-assignment s such that $\mathfrak{A}, s \models \alpha$.

In the above, it is always assumed that the structure $\mathfrak{A}$ gives meaning to all structure-symbols (predicates, functions, =) occurring in α and that s takes as values for the variables those which occur in the domain of $\mathfrak{A}$.

Corollary 22C. If S is a set of sentences and α is a sentence, then $S \models \alpha$ depends only on the structures involved and not on the s considered.

Example. Consider the following set S of axioms for a structure with equality and an edge relation E:

1. $\forall x \, \forall y \, [E(x, y) \to E(y, x)]$;

2. $\exists u \, \exists v \, \exists w \, \exists x \, \exists y \, \forall z \, [z = u \vee z = v \vee z = w \vee z = x \vee z = y]$;

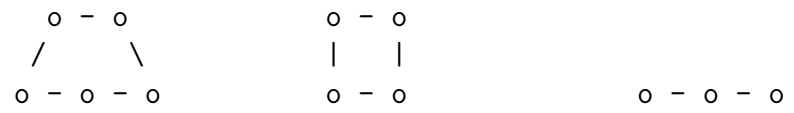
3. $\forall x \, [\neg E(x, x)]$;

4. $\forall x \, \forall y \, \exists z \, [(E(x, z) \wedge E(y, z)) \vee E(x, y)]$;

5. $\forall x \, \exists y \, \exists z \, \forall u \, [E(x, y) \wedge E(x, z) \wedge (E(x, u) \to (u = y \vee u = z))]$;

6. $\exists x \, \exists y \, [x \neq y \wedge \neg E(x, y)]$.

One can now draw three graphs which satisfy all axioms:



The first axiom says that the graph is undirected. The second axioms says that there are at most five vertices. The third axiom says that there are no self-loops. The fourth axiom says that any two vertices are either neighbours or have a common neighbour; the latter holds also when the two vertices are equal. The fourth axiom says that every vertex has at most two neighbours. Thus all graphs are either circles or lines with at most five vertices. The last axiom says that there are two vertices having the distance two, that is, they are not neighbours, but by the second axiom they have a common neighbour. For this to happen, one needs at least three vertices. Furthermore, as two distinct vertices cannot be neighbours, the triangle graph is not there. There is no line graph of four or more vertices, as then the end points are neither neighbours nor do they have a common neighbour. So, up to isomorphism, S has three models.

Example. Let the logical language just contain $=$ and a unary function symbol f and assume that the following axioms hold:

1. $\exists v\, \exists w\, \exists x\, \exists y\, \forall z\, [z = v \vee z = w \vee z = x \vee z = y]$;
2. $\forall x\, [f(f(f(f(x)))) = x]$.

Now one might ask how many structures are there satisfying these axioms. The following enumerated list gives them all.

1. Four models with $f(x) = x$ for all x and one through four elements;
2. Three models with two elements, a, b with $f(a) = b$ and $f(b) = a$ and zero, one or two further elements (named c, d when they exist) with $f(c) = c$ and $f(d) = d$;
3. One model with four elements a, b, c, d such that $f(a) = b$, $f(b) = a$, $f(c) = d$ and $f(d) = c$;
4. One model with four elements a, b, c, d such that $f(a) = b$, $f(b) = c$, $f(c) = d$ and $f(d) = a$.

This are nine models altogether. Note that a three-cycle ($f(a) = b, f(b) = c, f(c) = a$) cannot exist, as then $f(f(f(f(a)))) = b \neq a$. So the above list goes by "only one-cycles" in the first entry, "one two-cycle and perhaps some one-cycles" in the second entry, "two two-cycles" in the third entry and "one four-cycle" in the last entry. Note that by some convention in first-order logic, structures have always at least one element and therefore the empty set does not create a model here.

Example. One takes a structure as in the previous example, but in addition to the function f there are constants a, b, c, d and one requires that there are exactly four elements. So the following axioms hold:

1. $a \neq b \wedge a \neq c \wedge a \neq d \wedge b \neq c \wedge b \neq d \wedge c \neq d$;
2. $\forall x\, [x = a \vee x = b \vee x = c \vee x = d]$;
3. $\forall x\, [f(f(f(f(x)))) = x]$.

One would expect that there are less models as before, but that is not true. The reason is that two models can only be isomorphic when the same constants play the same role in both models. So if in one $f(a) = b$, the same must hold in the other one. Therefore one gets the following models:

1. One model with all cycles being one-cycles;
2. Six models with one two-cycle and two one-cycles, it is just the amount of two-element subsets in a four-element set;
3. Three models with two two-cycles, these are determined by choosing which value is $f(a)$, note that then $f(f(a)) = a$ and the other two elements also form one two-cycle;
4. Six models with one four-cycle, again these are determined by choosing $f(a)$ out of three choices and then $f(f(a))$ out of two remaining choices, $f(f(f(a)))$ has only one remaining choice and $f(f(f(f(a)))) = a$.

So one has in total sixteen models.

Definability in a Structure. A n-ary relation R is definable in a structure $\mathfrak{A}$ iff there is a first-order formula α with free variables $v_1, v_2, \ldots, v_n$ such that, for all

$a_1, a_2, \ldots, a_n \in A$, $R(a_1, a_2, \ldots, a_n)$ is true iff

$$\mathfrak{A}, s(v_1|a_1, v_2|a_2, \ldots, v_n|a_n) \models \alpha.$$

An n-ary function f is definable in a structure $\mathfrak{A}$ iff there is a first-order formula α with free variables $v_1, v_2, \ldots, v_n, v_{n+1}$ such that, for all structure elements $a_1, a_2, \ldots, a_n, b \in A$, $f(a_1, a_2, \ldots, a_n) = b$ iff

$$\mathfrak{A}, s(v_1|a_1, v_2|a_2, \ldots, v_n|a_n, v_{n+1}|b) \models \alpha.$$

An element c of the domain $\mathfrak{A}$ of a structure is definable iff there is a first-order formula α with one free variable v_1 such that for all $a \in A$,

$$\mathfrak{A}, s(v_1|a) \models \alpha \text{ if and only if } a = c.$$

The requirement that the first variables are used in the definition is not a real one, as one can rename the variables used in a formula.

Example. For example, the order is definable in $(\mathbb{N}, +)$ (with equality $=$) as $v_1 \leq v_2$ iff $\exists v_3 \, [v_1 + v_3 = v_2]$. Furthermore, every natural number n is definable in this structure. For example, n is that number for which there are exactly $n + 1$ pairs (x, y) with $x + y = n$. This can be formalised in first-order logic.

However, in the structure $(\mathbb{Z}, +)$, the order is not definable and also 0 is the only definable element by the formula $v_1 = v_1 + v_1$. So $0 = 0 + 0$ while, for example, $5 \neq 5 + 5$. This can be proven formally by showing that the mapping $a \mapsto -a$ is an isomorphism of the structure and as it maps non-zero numbers a to their negative counterpart, whenever a formula α is satisfiable with $v_1 = a$ then α is also satisfiable with $v_1 = -a$.

A mapping like $a \mapsto -a$ is an isomorphism in the structure $(\mathbb{Z}, +)$. If an element or a relation or a function is not preserved by some isomorphism of the structure onto itself then the corresponding element or relation or function is not definable. This will be made more precise later.

Example. Consider the graph

```
o - o - o
```

Now the middle element is definable by the formula

$$\exists y \, \exists w \, [E(x, y) \wedge E(x, z) \wedge y \neq z]$$

which is true iff x has at least two neighbours. However, the two elements on the ends of this line graph cannot be defined, as one can make a graph isomorphism of the graph onto itself which maps one end to the other end and vice versa. So one has one definable and two undefinable elements. Similarly, in the graph

```
o - o   o - o
```

none of the elements is definable while in the graph

```
o - o - o - o
    |   |   |
    o - o - o
```

every vertex is definable.

Definability of Classes of Structures. A semigroup is a structure with one binary operation $\circ$ which satisfies

$$\forall x \, \forall y \, \forall z \, [x \circ (y \circ z) = (x \circ y) \circ z]$$

and so one can define the class of all semigroups with this simple formula plus the constraint that the underlying language has equality and one binary operation symbol $\circ$, which could also be represented by a binary function f and the formula

$$\forall x \, \forall y \, \forall z \, [f(x, f(y, z)) = f(f(x, y), z)].$$

Of course, one might sometimes also use several or even infinitely many formulas to define a class of structures. A class of structures is definable iff there is a set S of axioms which defines it; some classes of structures are not definable like the class of all finite semigroups or all finite groups. These classes are nevertheless of interest for mathematicians.

Definition. A class C of structures is called "elementary" iff there is a single sentence α such that a structure belongs to C iff it satisfies the formula α; a structure is called "elementary in the wider sense" iff there is a set of sentences S such that a structure

belongs to C iff it satisfies this set of sentences. If possible, one tries to get that S is decidable or at least recursively enumerable; however, for some classes of structures this is impossible to achieve.

Example. The class of all groups is elementary and the class of all infinite groups is elementary in a wider sense. The class of all groups can be axiomatised by the following three sentences (which one can combine by $\wedge$ to get a single sentence):

$$\begin{aligned} &\forall x\, \forall y\, \forall z \quad [x \circ (y \circ z) = (x \circ y) \circ z]; \\ &\forall x\, \forall y\, \exists z \quad [x \circ z = y]; \\ &\forall x\, \forall y\, \exists z \quad [z \circ x = y]. \end{aligned}$$

The class of all infinite groups is augmented by formulas which enure, for every n, that there are at least n elements, that is, one quantifies existentially over $v_1, v_2, \ldots, v_n$ and then says that each v_i differs for v_j for i, j with $1 \leq i < j \leq n$. For example,

$$\exists v_1\, \exists v_2\, \exists v_3\, \exists v_4\, [v_1 \neq v_2 \wedge v_1 \neq v_3 \wedge v_1 \neq v_4 \wedge v_2 \neq v_3 \wedge v_2 \neq v_4 \wedge v_3 \neq v_4]$$

is the formula which says that there are at least four elements in the structure.

Relevant for Midterm Test up to This Point.

Homomorphisms and Isomorphisms. Assume that there are two structures $\mathfrak{A}$ and $\mathfrak{B}$ with the same language, that is, using the same relations and functions. For the ease of notation, also assume that the language contains the equality symbol $=$.

A mapping h from the domain A of $\mathfrak{A}$ to the domain B of $\mathfrak{B}$ is called a homomorphism iff it satisfies the following conditions:

1. For every constant symbol c, $h(c^{\mathfrak{A}}) = c^{\mathfrak{B}}$;
2. For every n-ary function symbol f and all $a_1, \ldots, a_n \in A$,
$$h(f^{\mathfrak{A}}(a_1, \ldots, a_n) = f^{\mathfrak{B}}(h(a_1), \ldots, h(a_n));$$
3. For every n-ary predicate symbol P and all $a_1, \ldots, a_n \in A$,
$$P^{\mathfrak{A}}(a_1, \ldots, a_n) \Rightarrow P^{\mathfrak{B}}(h(a_1), \ldots, h(a_n)).$$

A homomorphism h which satisfies in the third item for all predicates P and all $a_1, \ldots, a_n \in A$ the more restrictive condition

$$P^{\mathfrak{A}}(a_1, \ldots, a_n) \Leftrightarrow P^{\mathfrak{B}}(h(a_1), \ldots, h(a_n))$$

is called a *strong homomorphism*. A homomorphism is injective (= one-one) iff it furthermore satisfies $a = b \Leftrightarrow h(a) = h(b)$ for all $a, b \in A$ and a homomorphism is called an *isomorphism* iff it is a strong homomorphism, is injective and is surjective (= onto). The structures $\mathfrak{A}$ and $\mathfrak{B}$ are clled *isomorphic* iff they use the same logical language and there is an isomorphism from $\mathfrak{A}$ to $\mathfrak{B}$.

Homomorphism Theorem. Let $\mathfrak{A}$, $\mathfrak{B}$ be structures with the same logical language and let s be a mapping from the variables to the domain A of $\mathfrak{A}$. Let h be any homomorphism from A to B and let $s'(v_k) = h(s(v_k))$ for any variable v_k. Now the following statements hold.

(a) For every term t, $h(\overline{s}(t)) = \overline{s'}(t)$.

(b) For every atomic formula α, if $\mathfrak{A}, s \models \alpha$ then $\mathfrak{B}, s' \models \alpha$.

(c) If h is a strong homomorphism, then for every formula α without equality and without quantifiers, $\mathfrak{A}, s \models \alpha$ iff $\mathfrak{B}, s' \models \alpha$.

(d) If h is a strong injective homomorphism then for every formula α without quantifiers, $\mathfrak{A}, s \models \alpha$ iff $\mathfrak{B}, s' \models \alpha$.

(e) If h is a strong surjective homomorphism then for every formula α without equality, $\mathfrak{A}, s \models \alpha$ iff $\mathfrak{B}, s' \models \alpha$.

(f) If h is an isomorphism, then for every formula α, $\mathfrak{A}, s \models \alpha$ iff $\mathfrak{B}, s' \models \alpha$.

Proof. Now one proves all the different parts of the Homorphism Theorem.

(a) Note that $s'(v_k)$ maps variables to elements of the domain B of $\mathfrak{B}$ and that one can show by induction over the construction of terms that $h(\overline{s}(t)) = \overline{s'}(t)$. For the base case, it is easy to see that variables define this by definition and for constants, $h(\overline{s}(c)) = h(c^{\mathfrak{A}}) = c^{\mathfrak{B}} = \overline{s'}(c)$. Now, for induction, consider an n-ary function f and assume that the statement is already verified for the terms $t_1, \ldots, t_n$.

Now $h(\overline{s}(f(t_1,\ldots,t_n))) = h(f^{\mathfrak{A}}(\overline{s}(t_1),\ldots,\overline{s}(t_n))) = f^{\mathfrak{B}}(h(\overline{s}(t_1)),\ldots,h(\overline{s}(t_n))) = f^{\mathfrak{B}}(\overline{s'}(t_1)),\ldots,\overline{s'}(t_n)) = \overline{s'}(f(t_1,\ldots,t_n))$ and thus the statement also holds for the term $f(t_1,\ldots,t_n)$. Thus one can conclude by structural induction that the statement holds for all terms.

(b) If $\mathfrak{A}, s \models P(t_1,\ldots,t_n)$ then $P^{\mathfrak{A}}(\overline{s}(t_1,\ldots,\overline{s}(t_n))$ is true and $P^{\mathfrak{B}}(\overline{s'}(t_1),\ldots,\overline{s'}(t_n))$ is true and thus $\mathfrak{B}, s' \models P(t_1,\ldots,t_n)$. Furthermore, if $\mathfrak{A}, s \models t_1 = t_2$ then $\overline{s}(t_1) = \overline{s}(t_2)$ and $h(\overline{s}(t_1)) = h(\overline{s}(t_2))$ and $\overline{s'}(t_1) = \overline{s'}(t_2)$ and $\mathfrak{B}, s' \models t_1 = t_2$.

(c) For a strong homomorphism, which is obtained from atomic formulas consisting of predicates only using connectives, one can again show the formula by induction. For the base case, consider an n-ary predicate P with terms $t_1,\ldots,t_n$ and one sees that the following statements are all equivalent to each other: $\mathfrak{A}, s \models P(t_1,\ldots,t_n)$; $P^{\mathfrak{A}}(\overline{s}(t_1,\ldots,\overline{s}(t_n))$; $P^{\mathfrak{B}}(\overline{s'}(t_1),\ldots,\overline{s'}(t_n))$; $\mathfrak{B}, s' \models P(t_1,\ldots,t_n)$. Furthermore, one can see for the inductive step that whenever $\mathfrak{A}, s \models \alpha \Leftrightarrow \mathfrak{B}, s' \models \alpha$ and $\mathfrak{A}, s \models \beta \Leftrightarrow \mathfrak{B}, s' \models \beta$ then the same holds for $\alpha \to \beta$ and $\neg\alpha$.

(d) Assume now that h is an injective strong homomorphism. Using that $\mathfrak{A}, s \models t_1 = t_2$ iff $\overline{s}(t_1) = \overline{s}(t_2)$ and $\overline{s'}(t_1) = h(\overline{s}(t_1))$ and $\overline{s'}(t_2) = h(\overline{s}(t_2)$ it follows that $\mathfrak{A}, s \models t_1 = t_2$ iff $\overline{s'}(t_1) = \overline{s'}(t_2)$ and the latter is equivalent to $\mathfrak{B}, s' \models t_1 = t_2$. Using this as an additional item for the base case, the same inductive prove as before can be done to show that all quantifier-free formulas α satisfy that $\mathfrak{A}, s \models \alpha$ iff $\mathfrak{B}, s' \models \alpha$.

(e) Assume now that h is a surjective strong homomorphism. One extends now the proof of (c) to cover quantifiers. Here it is important that one does the induction in parallel for all s and derived s' and that one uses that every $s'(x|b)$ is derived from some $s(x|a)$ where a satisfies that $h(a) = b$. So in the following, assume that a satisfies $h(a) = b$ and assume that $\mathfrak{A}, s(x,a) \models \alpha$ iff $\mathfrak{B}, s'(x|h(a)) \models \alpha$. As for every b there is an a with $h(a) = b$, one can see that now the following two statements are equivalent:

- For all $a \in A$, $\mathfrak{A}, s(x|a) \models \alpha$ and
- For all $b \in B$, $\mathfrak{B}, s'(x|b) \models \alpha$.

The first of these statements is equivalent to $\mathfrak{A}, s \models \forall x\,[\alpha]$ and the second is equivalent to $\mathfrak{B}, s' \models \forall x\,[\alpha]$. This adjusts the proof of the inductive step missing in (c) in order to cover quantifiers and it can only be included when h is surjective.

(f) The full induction proof for the case of an isomorphism includes the proofs of (d) and (e) to be combined for those h which are both, injective and surjective.

This completes the proof. □

Elementary Equivalence. Two structures $\mathfrak{A}$ and $\mathfrak{B}$ are called *elementarily equivalent* iff they use the same logical language and for all sentences α, $\mathfrak{A} \models \alpha$ iff $\mathfrak{B} \models \alpha$.

Corollary 22D. Isomorphic structures are elementarily equivalent.

Example. The structures $(\mathbb{Q}, <)$ and $(\mathbb{R}, <)$ of the rational and real numbers are elementarily equivalent, but they are not isomorphic, as the rational and the real numbers have different cardinalities. This will be proven later as a corollary to the Theorem of Löwenheim and Skolem.

Corollary 22E. Let $\mathfrak{A}$ be any structure with domain A and R be any n-ary relation first-order definable in $\mathfrak{A}$ and h be any self-isomoprhism of $\mathfrak{A}$. Then, for the domain A of $\mathfrak{A}$ and any $a_1, \ldots, a_n \in A$, $R(a_1, \ldots, a_n)$ holds iff $R(h(a_1), \ldots, h(a_n))$ holds.

Applications of Corollary 22E. One can prove that certain relations, constants or functions are not definable by making isomorphisms which do not preserve the relations or constants or functions. One example is the structure $(\mathbb{Z}, +)$: In this structure, the mapping $z \mapsto -z$ is an isomorphism which does neither perserve the relation $<$ nor any constant different from 0; thus these constants and the order are not definable in the integers with addition.

On the other hand, the criterion is not an "if and only if" criterion. For example, every isomorphism of $(\mathbb{R}, +, \cdot)$ maps all integers to themselves, that is, cannot move any integer and also not move any rational. As $x \leq y \Leftrightarrow \exists z\,[x+z \cdot z = y]$, isomorphisms have to be order-preserving and there is only one isomorphism from $(\mathbb{R}, +, \cdot)$ to itself. This implies that every real r is moved to istelf by an isomorphism. However, there are only $\aleph_0$ many formulas and thus only $\aleph_0$ many real numbers are first-order definable in $(\mathbb{R}, +, \cdot)$, while there are $2^{\aleph_0}$ many reals. So many reals are neither moved by an isomorphism nor definable.

Chapter 2.4 – A Deductive Calculus

It is known that for sentential logic, the set of logical consequences of a recursively enumerable set of wffs is again recursively enumerable. For this and the next chapter, the same should be shown for first-order logic. More precisely, in this Chapter 2.4 a proof-calculus together with an algorithm to enumerate the formulas provable in this calculus is provided; in the next Chapter 2.5, a proof is provided, that provability in this calculus is also the same as being implied logically.

The Calculus. One calls a sequence $\beta_1, \beta_2, \ldots, \beta_n$ a proof for $S \models \alpha$ if some β_m is α and, for $m = 1, 2, \ldots, n$, the formula β_m is either a member of S or a member of Λ or there are $i, j < m$ with $\beta_i = (\beta_j \to \beta_m)$.

Explanation. The formulas $\beta_1, \beta_2, \ldots, \beta_n$ are considered to be steps of a logical derivation of α (and perhaps other formulas). In this derivation, each formula is either taken from S or is taken from the set of axioms Λ or is obtained by having derived two formulas β_j and $\beta_i = \beta_j \to \beta_m$ previously where the first formula establishes some fact and the second establishes that β_m is the consequence of this fact. This rule is called "Modus Ponens" which means if one has a formula β_j and a further formula $\beta_j \to \beta_m$ then one can put (ponere in Latin) the formula β_m. So all the intelligence of this proof method lies in the axiom set Λ and almost no intelligence in the rules which one is permitted to use.

The Axioms. The set Λ of axioms depends on the logical language used and is here given for a logical language with equality; if one does not have equality, axioms of type (5) and (6) are meaningless and can be dropped. The rules in Λ satisfy that they are valid, that is, they are true for every model and every s which uses the same logical language as the axioms. The axioms are taylormade for the connectives $\to$ and $\neg$; however, if one wants to use more axioms, one has to include the corresponding additional formulas into Λ in order to handle them.

(1) α for every α which is obtained by taking a tautology in sentential logic and replacing all atoms by well-formed formulas in a consistent way (the same atom needs always be replaced by the same formula);

(2) $\forall x\,[\alpha] \to (\alpha)^x_t$ for all well-formed formulas α, variables x and terms t where the substitution $(\alpha)^x_t$ is permitted, that is, it does not have any varialbe names inside t which are bound at the place x;

(3) $\forall x\,[\alpha \to \beta] \to \forall x\,[\alpha] \to \forall x\,[\beta]$;

(4) $\alpha \to \forall x\,[\alpha]$ for all well-formed formulas α and variables x where x does not occur free in α;

(5) $x = x$ for every variable x;

(6) $x = y \to \alpha \to \beta$ for all variables x, y and all atomic formulas α and all β derived from α by replacing some occurrences of x by occurrences of y or vice versa;

(7) $\forall x\,[\alpha]$ whenever α is in Λ by any of the previous axioms.

Here a substitution $(\alpha)^x_t$ replaces all free occurrences of x by the term t. Such a substitution is permitted only if it cannot create a contradiction. Here an example: $\exists y\,[y \neq x]$ is a formula which is true whenever the domain of the structure has at least two elements; however, when one substitutes x by y, one gets the formula $\exists y\,[y \neq y]$ which is not true in any structure. Thus the truth-value is compromised by transforming a free variable into a bound variable. Before defining when a substitution is permitted, here the formal definition of a substitution:

1. If α is atomic and t is a term, then $(\alpha)^t_x$ is obtained by replacing every occurrence of x by an occurrence of t; however, if x occurs in t itself, these new occurrences of x are *not* substituted in a chain-reaction;

2. $(\neg\alpha)^x_t$ is $(\neg\alpha^x_t)$;

3. $(\alpha \to \beta)^x_t$ is $(\alpha^x_t \to \beta^x_t)$.

4. $(\forall x\,[\alpha])^x_t$ is $\forall x\,[\alpha]$, that is, bound occurrences of x are not substituted;

5. $(\forall y\,[\alpha])^x_t$ is $\forall y\,[\alpha^x_t]$, where y is a variable different from x.

Furthermore, one defines as follows when a substitution is permitted:

1. If α is atomic then the substitution α^x_t is always permitted;

2. If α is of the form $(\neg\beta)$ or $(\beta \to \gamma)$ then α^x_t is permitted iff β^x_t and, in the case that it it is applicable, also γ^x_t are permitted;

3. If α is of the form $\forall y\,[\beta]$ then the substitution α_t^x is permitted iff either x does not occur free in α or y is a different variable than x and y does not occur in t and the substitution β_t^x is permitted.

Examples. In the following, let x, y, z be variables. Assume that the logical language contains $+$ and numerical constants (for making more interesting terms).

1. The substition $(\forall x\,\forall y\,[x = y \vee z \neq y])_y^x$ has no effect, whatever t is, and is therefore permitted.

2. The substition $(\forall x\,\forall y\,[x = y \vee z \neq y])_x^z$ produces the formula $\forall x\,\forall y\,[x = y \vee x \neq y]$ which is always true; this substitution is not permitted, as it replaces the free variable z by some term which is influenced by a quantified variable.

3. The substition $(\forall x\,\forall y\,[x = y \wedge z = y])_{x+1}^z$ produces the formula $\forall x\,\forall y\,[x = y \wedge x = y + 1]$ which is false in $\mathbb{N}$ and many other natural models. So a nonpermitted substitution can also make a formula false.

4. The substition $(\forall x\,\forall y\,[z \neq z{+}1])_{x+y}^z$ produces the formula $\forall x\,\forall y\,[x{+}y \neq x{+}y{+}1]$ and is not permitted, although it does not produce an obvious contradiction.

5. The substitution $((x \neq y \to x + 1 \neq y + 1)_8^x)_9^y$ provides the formula $8 \neq 9 \to 8 + 1 \neq 9_1$. The substition is permitted, as it is applied to an atomic formula.

Quiz. What are the results of the following substitutions and are they permitted?

1. $(\forall x\,[f(x) = y])_{x+8}^y$;
2. $(\forall x\,[f(x) = y])_{z+8}^y$.

Is the formula $(\forall x\,[\alpha]) \to \alpha$ in Λ? In all cases, in some cases (depending on α), in no case?

Notation of Proofs. One writes $S \vdash \beta$ for saying that one can derive or has derived the formula β from S using the formulas in S and the axioms Λ; so $\emptyset \vdash \beta$ means that β can be drieved from the axioms Λ alone. In general, if $S \subseteq S'$ then $S \vdash \beta$ implies $S' \vdash \beta$, as $S \vdash \beta$ only says that all the formulas copied in the proof are taken from S or Λ and as $S \subseteq S'$, those copied from S can also be copied from S'. However, one

normally tries to keep S as small as possible. The next example shows how to derive a quantifier free formula from S using the axioms for equality, namely the formula to be derived says that it does not matter in which order one writes the two sides of the equality sign.

Example of a Derivation in the Calculus. One wants to derive $\emptyset \vdash x = y \to y = x$, that is, that the equality is commutative as a relation and that this can be proven just using the axioms in Λ without any further precondition. Note that $(A_1 \to A_2 \to A_3) \to (A_2 \to A_1 \to A_3)$ is a tautology and this will be used below for a version of Axiom 1. Here, for convenience, "Axiom 1" just stands for "Axiom Group 1", as more precisely spoken, these axioms are a group of axioms and not just one. Similarly for the other axiom groups.

1. $\emptyset \vdash x = y \to x = x \to y = x$ (Axiom 6);
2. $\emptyset \vdash (x = y \to x = x \to y = x) \to (x = x \to x = y \to y = x)$ (Axiom 1);
3. $\emptyset \vdash x = x \to x = y \to y = x$ (Modus Ponens);
4. $\emptyset \vdash x = x$ (Axiom 5);
5. $\emptyset \vdash x = y \to y = x$ (Modus Ponens).

Theorem on Tautologies. If n formulas $\beta_1, \ldots, \beta_n$ tautologically imply α then $\{\beta_1, \ldots, \beta_n\} \vdash \alpha$.

Proof. The statement that "$\beta_1, \ldots, \beta_n$ tautologically imply α" is the same as saying that $(\beta_1 \wedge \ldots \wedge \beta_n) \to \alpha$ is a version of Axiom 1 and this is logically equivalent to $\beta_1 \to \ldots \to \beta_n \to \alpha$ which is in turn also a version of Axiom 1. So one can do the following derivation for $S = \{\beta_1, \ldots, \beta_n\}$

1. $S \vdash \beta_1 \to \ldots \to \beta_n \to \alpha$ (Axiom 1);
2. $S \vdash \beta_1$ (Copy)
3. $S \vdash \beta_2 \to \ldots \to \beta_n \to \alpha$ (Modus Ponens);
4. $S \vdash \beta_2$ (Copy)

5. $S \vdash \beta_3 \to \ldots \to \beta_n \to \alpha$ (Modus Ponens);
6. $S \vdash \beta_3$ (Copy)
7. $S \vdash \beta_4 \to \ldots \to \beta_n \to \alpha$ (Modus Ponens);
8. $\ldots$ $S \vdash \beta_n \to \alpha$ (Modus Ponens);
9. $S \vdash \beta_n$ (Copy);
10. $S \vdash \alpha$ (Modus Ponens).

This completes the proof. $\square$

Proof-Concatenation. If $S \vdash \beta$ for every $\beta \in S'$ and $S' \vdash \alpha$ then $S \vdash \alpha$. This statement can be proven by first determining all the formulas $\beta \in S'$ which are copied from S' and then replacing each such "copy and paste" by the proof for $S \vdash \beta$. The resulting longer proof is actually a proof for $S \vdash \alpha$. This principle gives the following corollary.

Corollary. If $S \vdash \beta_1, \ldots, S \vdash \beta_n$ and $\beta_1, \ldots, \beta_n$ tautologically imply α then $S \vdash \alpha$.

Abbreviated Notation. For the following, it should be allowed to write

1. $\forall x\, [\alpha]$ (....);
2. α_t^x (Axiom 2, $x \to t$, Modus Ponens);

for the sequence

1. $\forall x\, [\alpha]$ (....);
2. $\forall x\, [\alpha] \to \alpha_t^x$ (Axiom 2);
3. α_t^x (Modus Ponens);

in order to avoid writing lengthy formulas as produced by Axiom 2. So, for example, the lines

1. $S \vdash \forall y\, \forall x\, [x + y = y + x]$ (Copy);

2. $S \vdash \forall x\,[x + 0 = 0 + x]$ (Axiom 2, $y \to 0$, Modus Ponens);

in the below proof stand for

1. $S \vdash \forall y\,\forall x\,[x + y = y + x]$ (Copy);
2. $S \vdash \forall y\,\forall x\,[x + y = y + x] \to \forall x\,[x + 0 = 0 + x]$ (Axiom 2);
3. $S \vdash \forall x\,[x + 0 = 0 + x]$ (Modus Ponens);

and it is easy to see that this will save some writing-time, as this sequence of proof-steps is quite frequent.

Example of a Derivation in the Calculus. One wants to derive that $S = \{\forall y\,\forall x\,[x + y = y + x], \forall x\,[x + 0 = x]\}$ proves $0 + 5 = 5$.

1. $S \vdash \forall y\,\forall x\,[x + y = y + x]$ (Copy);
2. $S \vdash \forall x\,[x + 0 = 0 + x]$ (Axiom 2, $y \to 0$, Modus Ponens);
3. $S \vdash 5 + 0 = 0 + 5$ (Axiom 2, $x \to 5$, Modus Ponens);
4. $S \vdash \forall v\,\forall w\,\forall u\,[v = w \to v = u \to w = u]$ (Axiom 6,7);
5. $S \vdash \forall w\,\forall u\,[5 + 0 = w \to 5 + 0 = u \to w = u]$ (Axiom 2, $v \to 5 + 0$, Modus Ponens);
6. $S \vdash \forall u\,[5 + 0 = 0 + 5 \to 5 + 0 = u \to 0 + 5 = u]$ (Axiom 2, $w \to 5 + 0$, Modus Ponens);
7. $S \vdash 5 + 0 = 0 + 5 \to 5 + 0 = 5 \to 0 + 5 = 5$ (Axiom 2, $u \to 5$, Modus Ponens);
8. $S \vdash 5 + 0 = 5 \to 0 + 5 = 5$ (Modus Ponens);
9. $S \vdash \forall x\,[x + 0 = x]$ (Copy);
10. $S \vdash 5 + 0 = 5$ (Axiom 2, $x \to 5$, Modus Ponens);
11. $S \vdash 0 + 5 = 5$ (Modus Ponens).

Deduction Theorem. If $S \cup \{\alpha\} \vdash \beta$ then $S \vdash \alpha \to \beta$.

Proof. The idea is that a formal proof of $S \cup \{\alpha\} \vdash \beta$ consisting of steps $\gamma_1, \gamma_2, \ldots, \gamma_n$ with $\beta = \gamma_n$ can be translated into a formal proof such that each step

- $S \cup \{\alpha\} \vdash \gamma_m$;

becomes translated into one or more steps with the last one of the new steps being

- $S \vdash \alpha \to \gamma_m$;

and this is now more detailed done in a case distinction. In the case that $\gamma_m \in S$, the complete step is as follows:

- $S \vdash \gamma_m$ (Copy);
- $S \vdash \gamma_m \to \alpha \to \gamma_m$ (Axiom 1);
- $S \vdash \alpha \to \gamma_m$ (Modus Ponens);

here one has to know that $A_1 \to A_2 \to A_1$ is a tautology, as it is equivalent to $(A_1 \wedge A_2) \to A_1$. In the case that γ_m equals α, one just puts the following single derivation step:

- $S \vdash \alpha \to \alpha$ (Axiom 1);

and this is based on the fact that $A_1 \to A_1$ is always true in propositional logic. In the case that γ_m is in Λ, more specifically a version of Axiom k with $k \in \{1, 2, 3, 4, 5, 6, 7\}$, one puts the following steps:

- $S \vdash \gamma_m$ (Axiom k);
- $S \vdash \gamma_m \to \alpha \to \gamma_m$ (Axiom 1);
- $S \vdash \alpha \to \gamma_m$ (Modus Ponens);

and these are essentially equal to copying from S. The last case is that of a Modus Ponens. Originally one derives γ_m from previous steps γ_i and $\gamma_j = \gamma_i \to \gamma_m$; now one has instead $\alpha \to \gamma_i$ and $\alpha \to \gamma_i \to \gamma_m$. So one makes use of the following tautology: $(A_1 \to A_2 \to A_3) \to (A_1 \to A_2) \to (A_1 \to A_3)$. Note that $(A_1 \to A_2) \to (A_1 \to A_3)$ can only be false when $(A_1 \to A_3)$ is false and $(A_1 \to A_2)$ is true; so this can only be when A_1 is true, A_2 is true and A_3 is false. But in this case, also $A_1 \to A_2 \to A_3$ is false, thus the overall implication is true. For that reason, one can now put the following steps into the derivation:

- $S \vdash (\alpha \to \gamma_i \to \gamma_m) \to (\alpha \to \gamma_i) \to (\alpha \to \gamma_m)$ (Axiom 1);
- $S \vdash (\alpha \to \gamma_i) \to (\alpha \to \gamma_m)$ (Modus Ponens);
- $S \vdash \alpha \to \gamma_m$ (Modus Ponens);

here the first Modus Ponens step used that $\alpha \to \gamma_i \to \gamma_m$ occurred earlier in the derivation and the second Modus Ponens step used that $\alpha \to \gamma_i$ occurred earlier in the derivation. Thus this allows to translate the usage of Modus Ponens. Thus every step can be translated and the new formal proof is for $S \vdash \alpha \to \beta$. $\square$
If $S \vdash \alpha \to \beta$ then

1. $S \cup \{\alpha\} \vdash \alpha \to \beta$ (Superset of S);
2. $S \cup \{\alpha\} \vdash \alpha$ (Copy);
3. $S \cup \{\alpha\} \vdash \beta$ (Modus Ponens).

Thus one can generalise the Deduction Theorem in order to make it if and only if.

Deduction Theorem. $S \cup \{\alpha\} \vdash \beta$ iff $S \vdash \alpha \to \beta$.

Proof by Contraposition. $S \cup \{\alpha\} \vdash \beta$ if and only if $S \cup \{\neg\beta\} \vdash \neg\alpha$.

Proof. This proof needs the usage of the Deduction Theorem.

1. $S \cup \{\alpha\} \vdash \beta$ (Assumption);
2. $S \vdash \alpha \to \beta$ (Deduction Theorem);
3. $S \vdash (\alpha \to \beta) \to (\neg\beta \to \neg\alpha)$ (Axiom 1);
4. $S \vdash (\neg\beta \to \neg\alpha)$ (Modus Ponens);
5. $S \cup \{\neg\beta\} \vdash \neg\alpha$ (Deduction Theorem).

Now one does the opposite direction. Here one uses an additional version of the Axiom 1 which allows to replace $\neg\neg\alpha$ by α and $\neg\neg\beta$ by β in the given formula.

1. $S \cup \{\neg\beta\} \vdash \neg\alpha$ (Assumption);

2. $S \vdash (\neg\beta \to \neg\alpha)$ (Deduction Theorem);

3. $S \vdash (\neg\beta \to \neg\alpha) \to (\neg\neg\alpha \to \neg\neg\beta)$ (Axiom 1);

4. $S \vdash \neg\neg\alpha \to \neg\neg\beta$ (Modus Ponens);

5. $S \vdash (\neg\neg\alpha \to \neg\neg\beta) \to (\alpha \to \beta)$ (Axiom 1);

6. $S \vdash \alpha \to \beta$ (Modus Ponens);

7. $S \cup \{\alpha\} \vdash \beta$ (Deduction Theorem).

Note that one could also have combined the two usages of Axiom 1 in this direction to one singular, slightly more complicated version of these two axioms. □

Definition. One says that a set S of formulas is *inconsistent*, if one can derive an antitautology. Equivalently, one can also say that S is *inconsistent* if one can derive, for some β, both the formulas β and $\neg\beta$. Another equivalent definition is that S is *inconsistent* iff every formula can be derived from S. This statement is a consequence of the fact that for every α, the formula $\beta \to \neg\beta \to \alpha$ is a tautology; this is most easily seen by its equivalent form $(\beta \wedge \neg\beta) \to \alpha$. So when one can derive β and $\neg\beta$, then one can also in two further steps derive α.

Reductio ad Absurdum. If $S \cup \{\alpha\}$ is inconsistent then $S \vdash \neg\alpha$.

Proof. This can be shown by the following proof.

1. $S \cup \{\alpha\} \vdash \neg\alpha$ (Assumption);

2. $S \vdash \alpha \to \neg\alpha$ (Deduction Theorem);

3. $S \vdash (\alpha \to \neg\alpha) \to \neg\alpha$ (Axiom 1);

4. $S \vdash \neg\alpha$ (Modus Ponens).

Note that $\alpha \to \neg\alpha$ is true iff $\neg\alpha$ is true; hence $(\alpha \to \neg\alpha) \to \neg\alpha$ is a tautology and so can be generated by Axiom 1. The reason is that if $\neg\alpha$ is true, then the statement is true; if $\neg\alpha$ is false, then α is true and the precondition $\alpha \to \neg\alpha$ is false, so that the overall implication is again true. Note that the usage of Modus Ponens needs the brackets as placed in the formulas above. □

Generalisation Theorem. If $S \vdash \alpha$ and the variable x does not occur free in S then $S \vdash \forall x\,[\alpha]$. Again one proves that by translating the proof. Assume that there is a sequence of formulas $\beta_1, \beta_2, \ldots, \beta_n$ which is a proof for α from S and $\alpha = \beta_n$. Now one replaces every single step

- $S \vdash \beta_m$;

by a sequence of steps such that the last of these is

- $S \vdash \forall x\,[\beta_m]$;

and the choice of the statements depends on how β_m is derived. If $\beta_m \in \Lambda$ by Axiom k then one directly replace this by

- $S \vdash \forall x\,[\beta_m]$ (Axiom k and Axiom 7);

and nothing more needs to be proven. The next case is that $\beta_m \in S$. As x does not occur free in β_m, one can then use Axiom 4.

- $S \vdash \beta_m$ (Copy);
- $S \vdash \beta_m \to \forall x\,[\beta_m]$ (Axiom 4);
- $S \vdash \forall x\,[\beta_m]$ (Modus Ponens);

here the Axiom 4 is needed and the proof of the Generalisation Theorem is more or less the only place where one needs it; later one will just apply the Generalisation Theorem instead. The last case is that there are $i, j < m$ with $\beta_j = \beta_i \to \beta_m$ and the application of the Modus Ponens in the original proof. This translates as follows.

- $S \vdash \forall x\,[\beta_i]$ (Somewhere above in proof);
- $S \vdash \forall x\,[\beta_i \to \beta_m]$ (Somewhere above in proof);
- $S \vdash \forall x\,[\beta_i \to \beta_m] \to \forall x\,[\beta_i] \to \forall x\,[\beta_m]$ (Axiom 3);
- $S \vdash \forall x\,[\beta_i] \to \forall x\,[\beta_m]$ (Modus Ponens);
- $S \vdash \forall x\,[\beta_m]$ (Modus Ponens);

here recall that Axiom 3 was stating that one can move a quantifier from outside over an implication. Only the last three steps are the new part for β_m, the other two steps were only cited to explain why the Modus Ponens can be used twice. The so translated proof is then able to derive $\forall x\,[\alpha]$ from S. $\square$

Generalisation of Constants. Assume that c is a constant which does not appear in S and $S \vdash \alpha$. Now there is a variable y which does not appear in α such that $S \vdash \forall y\,[\alpha_y^c]$. Furthermore, there is a deduction of this statement which does not use the constant c at all.

Proof. Let $\beta_1, \beta_2, \ldots, \beta_n$ be a proof for α using S and $\beta_n = \alpha$. Now $\{\beta_1, \ldots, \beta_n\}$ is finite and thus there is a variable y which does not occur in any of these formulas, in particular it holds that $T = S \cap \{\beta_1, \ldots, \beta_n\}$ satisfies $T \vdash \alpha$. Now it is proven that $(\beta_1)_y^c, (\beta_2)_y^c, \ldots, (\beta_n)_y^c$ is a proof for α_y^c. For each m, there are several cases:

- If $\beta_m \in S$ then β_m does not contain c and $(\beta_m)_y^c = \beta_m$, so β_m can still be copied from S;
- If $\beta_m \in \Lambda$ then one can see that the new axiom $(\beta_m)_y^c$ can also be obtained that way and that no conflict is introduced for versions of Axiom 2, as y does not occur in bound form in β_m by assumption;
- If β_m is obtained by Modus Ponens from β_i and $\beta_j = \beta_i \to \beta_m$, then in the new proof, $(\beta_m)_y^c$ is obtained by Modus Ponens from $(\beta_i)_y^c$ and $(\beta_j)_y^c$, as $(\beta_j)_y^c = (\beta_i)_y^c \to (\beta_m)_y^c$.

Furthermore, as T does not contain the variable y free, it follows by the Generalisation Theorem that there is also a proof for $T \vdash \forall y\,[\alpha_y^c]$ and this proof does not reintroduce the constant c, as it does not occur in T. As $S \supseteq T$, it follows that $S \vdash \forall y\,[\alpha_y^c]$ by the same proof. $\square$

Corollary 24G. Assume that $S \vdash \alpha_c^x$, where the constant symbol c neither occurs in S nor in α. Then $S \vdash \forall x\,[\alpha]$ and there is a deduction of this formula which does not use the constant symbol c.

Proof. Assume that the formal proof for $S \vdash \alpha_c^x$ is given and y is a variable which does not occur in any formula of this proof. Now there is also a proof for $S \vdash (\alpha_c^x)_y^c$ which is, more precisely, a proof for $S \vdash \alpha_y^x$. This proof can be translated into a

proof for $S \vdash \forall y\,[\alpha_y^x]$. As x does not occur free in α_y^x, $(\alpha_y^x)_x^y$ is the same as α and furthermore, y occurs in α_y^x only in places where x is free (as otherwise the y would not have gone there at the substitution α_y^x). So the substitution is permitted and the formula $\forall y\,[\alpha_y^x] \to \alpha$ is a version of Axiom 2. Thus one can deduce $\forall y\,[\alpha_y^x] \to \alpha$ and similarly $\forall x\,[\forall y\,[\alpha_y^x] \to \alpha]$ is also an Axiom in Λ. So one has the following end of the derivation:

1. $S \vdash \forall x\,[\forall y\,[\alpha_y^x] \to \alpha]$ (as explained above);
2. $S \vdash \forall x\,[\forall y\,[\alpha_y^x]] \to \forall x\,[\alpha]$ (Axiom 3 and Modus Ponens);
3. $S \vdash \forall y\,[\alpha_y^x] \to \forall x\,\forall y\,[\alpha_y^x]$ (Axiom 4);
4. $S \vdash \forall x\,\forall y\,[\alpha_y^x]$ (Modus Ponens);
5. $S \vdash \forall x\,[\alpha]$ (Modus Ponens).

Here the third step just used that x is not free in the formula $\forall y\,[\alpha_y^x]$, as all free occurrences of x in α had been renamed to y. □

Corollary 24H. Assume that a constant c does not occur in the formulas S, α and β. If $S \cup \{\alpha_c^x\} \vdash \beta$ then $S \cup \exists x\,[\alpha]\} \vdash \beta$ and there is a formal proof for the latter which does not use the constant c.

This rule is known as "Existential Instantiation".

Proof. This result builds on Corollary 24G and Contraposition.

1. $S \cup \{\alpha_c^x\} \vdash \beta$ (Assumption);
2. $S \cup \{\neg\beta\} \vdash \neg\alpha_c^x$ (Contraposition);
3. $S \cup \{\neg\beta\} \vdash \forall x\,[\neg\alpha]$ (Corollary 24G);
4. $S \cup \{\exists x\,[\alpha]\} \vdash \beta$ (Contraposition).

Note that the statement $S \cup \{\neg\beta\} \vdash \forall x\,[\neg\alpha]$ does no longer contain c on either side and therefore one can, by Corollary 24G, obtain a proof for this fact without using the constant c; this proof can then be extended by the last steps to incorporate the Contraposition. □

Alphabetic Variants. For every formula α and every term t there is a formula β which is obtained from α by renaming the bound variables consistently such that no bound variable of β occurs in t and β is provably equal to α, that is, the substitution β_t^x is permitted and $\emptyset \vdash \alpha \to \beta$ and $\emptyset \vdash \beta \to \alpha$.

Proof. One proves this for each fixed term t and variable x by structural induction over α; for this one defines recursively a function F from wffs to wffs (which depends on t) such that $\beta = F(\alpha)$ is the corresponding formula and this formula satisfies the following invariants:

1. No bound variable in $F(\alpha)$ occurs in t and thus the substitution $(F(\alpha))_t^x$ is permitted;
2. $\emptyset \vdash \alpha \to F(\alpha)$;
3. $\emptyset \vdash F(\alpha) \to \alpha$.

First, for atomic formulas, one defines $F(\alpha) = \alpha$ and there are no bound variables at all. The substitution $(F(\alpha))_t^x$ is therefore permitted. Furthermore, as $\alpha = F(\alpha)$, the equivalence of α and $F(\alpha)$ is provable.

If $\alpha = \gamma \to \delta$ then $F(\alpha) = F(\gamma) \to F(\delta)$ or $\neg F(\gamma)$, respectively. By induction hypothesis, all bound variables in $F(\gamma)$ and $F(\delta)$ do not occur in t and the same holds for $F(\alpha)$. Thus the substitution $(F(\alpha))_t^x$ is permitted. Furthermore, by assumption, $\emptyset \vdash \gamma \to F(\gamma)$, $\emptyset \vdash F(\gamma) \to \gamma$, $\emptyset \vdash \delta \to F(\delta)$, $\emptyset \vdash F(\delta) \to \delta$. Now the following two formulas are tautologies, which say that when γ and $F(\gamma)$ are equivalent and when δ and $F(\delta)$ are equivalent then $(\gamma \to \delta) \to (F(\gamma) \to F(\delta))$ and the other way round:

- $(\gamma \to F(\gamma)) \to (F(\gamma) \to \gamma) \to (\delta \to F(\delta)) \to (F(\delta) \to \delta) \to ((\gamma \to \delta) \to (F(\gamma) \to F(\delta)))$;
- $(\gamma \to F(\gamma)) \to (F(\gamma) \to \gamma) \to (\delta \to F(\delta)) \to (F(\delta) \to \delta) \to ((F(\gamma) \to F(\delta)) \to (\gamma \to \delta))$.

Using these formulas by Axiom 1 and then applying Modus Ponens four times, one can derive the two formulas $\alpha \to F(\alpha)$ and $F(\alpha) \to \alpha$ and these are valid.

If $\alpha = \neg\gamma$, then one let $F(\alpha) = \neg F(\gamma)$. As no bound variable of γ occurs in t, the same is true for $\alpha = \neg\gamma$ and therefore the substitution α_t^x is permitted. Using Contraposition, one gets out of $\gamma \to F(\gamma)$ and $F(\gamma) \to \gamma$ the formulas $\neg F(\gamma) \to \neg\gamma$

and $\neg\gamma \rightarrow \neg F(\gamma)$; as the two first were valid by induction hypothesis, the two last are valid as well.

If $\alpha = \forall y\,[\gamma]$, then let z be any variable which neither occurs in $F(\gamma)$ nor in t; as there are infinitely many variables, such a variable z exists. Now one lets $F(\alpha) = \forall z\,[(F(\gamma))^y_z]$. Note that therefore no bound variable of $F(\alpha)$ occurs in t, as this holds for γ by induction hypothesis and the only new quantified variable is z which does not occur in t. Furthermore, $(F(\gamma))^y_z$ is a permitted substitution and note that it only affects those occurrences of y in $F(\gamma)$ where y occurs free in $F(\gamma)$. Note that y, if it occurs free in γ, also occurs at the corresponding places free in $F(\gamma)$, as F does nothing else than changing the names of quantified variables within the range of the corresponding quantifiers. Furthermore, the substition $(\forall z\,[(F(\gamma))^y_z])^x_t$ is also permitted, as none of the bound variables in this formula occurs in t. It remains to show that the proven equivalence of the new formulas goes through and this is done by making the corresponding formal proofs.

1. $\emptyset \vdash \gamma \rightarrow F(\gamma)$ (Induction Hypothesis);
2. $\emptyset \vdash \forall y\,[\gamma \rightarrow F(\gamma)]$ (Generalisation Theorem);
3. $\emptyset \vdash \forall y\,[\gamma] \rightarrow \forall y\,[F(\gamma)]$ (Axiom 3, Modus Ponens);
4. $\{\forall y\,[\gamma]\} \vdash \forall y\,[F(\gamma)]$ (Deduction Theorem);
5. $\{\forall y\,[\gamma]\} \vdash \forall y\,[F(\gamma)] \rightarrow (F(\gamma))^y_z$ (Axiom 2);
6. $\{\forall y\,[\gamma]\} \vdash (F(\gamma))^y_z$ (Modus Ponens);
7. $\{\forall y\,[\gamma]\} \vdash \forall z\,[(F(\gamma))^y_z]$ (Generalisation Theorem);
8. $\{\alpha\} \vdash F(\alpha)$ (Same as Before);
9. $\emptyset \vdash \alpha \rightarrow F(\alpha)$ (Deduction Theorem).

For the converse direction, one uses that all occurrences of z in $(F(\gamma))^y_z)$ had been places where y occurrs free and thus $((F(\gamma))^y_z)^z_y$ just equals $F(\gamma)$.

1. $\emptyset \vdash \forall z\,[(F(\gamma))^y_z] \rightarrow ((F(\gamma)^z_y)^z_y$ (Axiom 2);
2. $\{\forall z\,[(F(\gamma))^y_z]\} \vdash ((F(\gamma)^y_z)^z_y$ (Deduction Theorem);

3. $\{F(\alpha)\} \vdash F(\gamma)$ (The same written differently);

4. $\{F(\alpha)\} \vdash F(\gamma) \to \gamma$ (Induction Hypothesis);

5. $\{F(\alpha)\} \vdash \gamma$ (Modus Ponens);

6. $\{F(\alpha)\} \vdash \forall y\,[\gamma]$ (Generalisation Theorem);

7. $\{F(\alpha)\} \vdash \alpha$ (The same written differently);

8. $\emptyset \vdash F(\alpha) \to \alpha$ (Deduction Theorem).

This completes the proof. □

Example. Alphabetic variants are employed to avoid conflicts with bound variables; the following example gives some insight.

Let t be the term $x \circ y$ and consider the following formula:

$$\forall z\,\forall x\,\forall y\,[(x \circ y) \circ z = z]$$

and one wants to substitute z by $x \circ y$. This would clash with the bound variables x and y and thus it cannot be done. However, one can form the alphabetical variant

$$\forall z\,\forall v\,\forall w\,[(v \circ w) \circ z = z]$$

and obtains from this using Axiom 2 and Modus Ponens the formula

$$\forall v\,\forall w\,[(v \circ w) \circ (x \circ y) = x \circ y]$$

A structure here where the above law holds is any semigroup where the semigroup operation is defined by the equation $x \circ y = y$ for all x, y. It is associative, but does not have a neutral element, so it is not a monoid.

Example of Deriviations. The following formulas are proven using the axioms, the Deduction Theorem and the Generalisation Theorem. All the results proven will concern the relate equality with a function symbol. First one proves the valid formula

$$\forall x\,\forall y\,[f(x) = f(y) \to f(y) = f(x)]$$

where f is a unary function symbol. The formal derivation is the following:

1. $\emptyset \vdash v = w \to v = v \to w = v$ (Axiom 6);
2. $\{v = w\} \vdash v = v \to w = v$ (Deduction Theorem);
3. $\{v = w\} \vdash v = v$ (Axiom 5);
4. $\{v = w\} \vdash w = v$ (Modus Ponens);
5. $\emptyset \vdash v = w \to w = v$ (Deduction Theorem);
6. $\emptyset \vdash \forall w\,[v = w \to w = v]$ (Generalisation Theorem);
7. $\emptyset \vdash \forall v\,\forall w\,[v = w \to w = v]$ (Generalisation Theorem);
8. $\emptyset \vdash \forall w\,[f(x) = w \to w = f(x)]$ (Axiom 2, v replaced by $f(x)$, Modus Ponens);
9. $\emptyset \vdash f(x) = f(y) \to f(y) = f(x)$ (Axiom 2, w replaced by $f(y)$, Modus Ponens);
10. $\emptyset \vdash \forall y\,[f(x) = f(y) \to f(y) = f(x)]$ (Generalisation Theorem);
11. $\emptyset \vdash \forall x\,\forall y\,[f(x) = f(y) \to f(y) = f(x)]$ (Generalisation Theorem).

A further valid formula is

$$\forall x\,\forall y\,[x = y \to f(x) = f(y)]$$

and the proof is similar, though not identical:

1. $\emptyset \vdash x = y \to f(x) = f(x) \to f(x) = f(y)$ (Axiom 6);
2. $\{x = y\} \vdash f(x) = f(x) \to f(x) = f(y)$ (Deduction Theorem);
3. $\{x = y\} \vdash \forall z\,[z = z]$ (Axiom 5);
4. $\{x = y\} \vdash \forall z\,[z = z] \to f(x) = f(x)$ (Axiom 2);
5. $\{x = y\} \vdash f(x) = f(x)$ (Modus Ponens);
6. $\{x = y\} \vdash f(x) = f(y)$ (Modus Ponens);
7. $\emptyset \vdash x = y \to f(x) = f(y)$ (Deduction Theorem);

8. $\emptyset \vdash \forall y\,[x = y \to f(x) = f(y)]$ (Generalisation Theorem);

9. $\emptyset \vdash \forall x\,\forall y\,[x = y \to f(x) = f(y)]$ (Generalisation Theorem).

The next statement says that if f is one-one, so is the concatenation of f with itself:

$$\{\forall x\,\forall y\,[x \neq y \to f(x) \neq f(y)]\} \vdash \forall x\,\forall y\,[x \neq y \to f(f(x)) \neq f(f(y))].$$

To simplify the notation, let α stand for $x \neq y \to f(x) \neq f(y)$. The proof is now the following:

1. $\{\forall x\,\forall y\,[\alpha]\} \vdash \forall x\,\forall y\,[\alpha]$ (Copy);

2. $\{\forall x\,\forall y\,[\alpha]\} \vdash \forall y\,[\alpha]$ (Axiom 2, no replacement, Modus Ponens);

3. $\{\forall x\,\forall y\,[\alpha]\} \vdash \alpha$ (Axiom 2, no replacement, Modus Ponens);

4. $\{\forall x\,\forall y\,[\alpha]\} \vdash \forall y\,[f(x) \neq y \to f(f(x)) \neq f(y)]$ (Axiom 2 on first formula, replacement x by $f(x)$, Modus Ponens);

5. $\{\forall x\,\forall y\,[\alpha]\} \vdash f(x) \neq f(y) \to f(f(x)) \neq f(f(y))$ (Axiom 2, replacement y by $f(y)$, Modus Ponens);

6. $\{\forall x\,\forall y\,[\alpha]\} \vdash (x \neq y \to f(x) \neq f(y)) \to (f(x) \neq f(y) \to f(f(x)) \neq f(f(y))) \to (x \neq y \to f(f(x)) \neq f(f(y)))$ (Axiom 1, Transitivity of $\to$);

7. $\{\forall x\,\forall y\,[\alpha]\} \vdash (f(x) \neq f(y) \to f(f(x)) \neq f(f(y))) \to (x \neq y \to f(f(x)) \neq f(f(y)))$ (Modus Ponens);

8. $\{\forall x\,\forall y\,[\alpha]\} \vdash x \neq y \to f(f(x)) \neq f(f(y))$ (Modus Ponens);

9. $\{\forall x\,\forall y\,[\alpha]\} \vdash \forall y\,[x \neq y \to f(f(x)) \neq f(f(y))]$ (Generalisation Theorem);

10. $\{\forall x\,\forall y\,[\alpha]\} \vdash \forall x\,\forall y\,[x \neq y \to f(f(x)) \neq f(f(y))]$ (Generalisation Theorem).

The transitivity of $\to$ is the formula which says that when $A_1 \to A_2$ and $A_2 \to A_3$ then $A_1 \to A_3$. That is, one takes the formula $(A_1 \to A_2) \to (A_2 \to A_3) \to (A_1 \to A_3)$ and replaces A_1 by $x \neq y$, A_2 by $f(x) \neq f(y)$ and A_3 by $f(f(x)) \neq f(f(y))$. The last two invocations of the Generalisation Theorem can be applied, as x and y do not occur free in $\forall x\,\forall y\,[\alpha]$.

Chapter 2.5 – Soundness and Completeness Theorems

Soundness and Completeness. A proof-system is called *sound* iff it only proves correct theorems, that is, whenever $S \vdash \alpha$ in that proof system then also $S \models \alpha$. A proof-system is called *complete* iff it proves every correct theorem, that is, whenever $S \models \alpha$ then $S \vdash \alpha$ in this system. A fundamental result for first-order logic is that the proof system from Chapter 2.4 is both, sound and complete. Furthermore, it can be carried out by algorithms (though these need on many provable theorems an excessive amount of computation time and space).

Lemma. If a formula α is valid, then also $\forall x\,[\alpha]$.

Proof. Assume that α is valid, that is, for every structure $\mathfrak{A}$ with some domain A and every s mapping variables to elements of A,

$$\mathfrak{A}, s \models \alpha.$$

Now, the same holds if one modifies s at x to some $a \in A$: For all s and all value $a \in A$,

$$\mathfrak{A}, s(x|a) \models \alpha.$$

It follows by the definition of the universal quantifier that for all structures $\mathfrak{A}$ and all s,

$$\mathfrak{A}, s \models \forall x\,[\alpha];$$

thus $\forall x\,[\alpha]$ is also a valid formula. □

Soundness of Λ. For the soundness of the Proof Calculus from Chapter 2.4, it is sufficient to show Axioms 1 – 6, as Axiom 7 follows from the preceding lemma.

Axiom 1 considers formulas which are tautological combinations of easier formulas. That is, given a formula like

$$\alpha \to \beta \to \alpha,$$

the truth-value of this formula in a structure depends only on what the structure does with α and β, so let $\nu(A_1) = 1$ iff $\mathfrak{A}, s \models \alpha$ and $\nu(A_2) = 1$ iff $\mathfrak{A}, s \models \beta$. Now one has that

$$\mathfrak{A}, s \models \alpha \to \beta \to \alpha$$

iff $\overline{\nu}(A_1 \to A_2 \to A_1) = 1$; as this underlying formula is a tautology, this is always true. Thus the formula

$$\alpha \to \beta \to \alpha$$

is valid, regardless of what α and β are, as long as they are wffs.

Axiom 2 considers formulas which are of the form $\forall x\,[\alpha] \to \alpha_t^x$, where this substitution is permitted. Now if $\mathfrak{A}, s \models \forall x\,[\alpha]$, then $\mathfrak{A}, s$ makes α_a^x for every value $a \in A$ true. This is true in particular for the value $a = \overline{s}(t)$ and this value is at all places where x occurs not modified by bound variables of α, as otherwise the substitution would not be permitted. Thus if $\mathfrak{A}, s \models \forall x\,[\alpha]$ then $\mathfrak{A}, s \models \alpha_t^x$. This implication is then a valid formula: $\mathfrak{A}, s \models \forall x\,[\alpha] \to \alpha_t^x$. As every $\mathfrak{A}$ can be chosen which uses all logical symbols appearing in α and every mapping s from variables to members of the domain of $\mathfrak{A}$ can be taken, the formula $\forall x\,[\alpha] \to \alpha_t^x$ is then valid.

Axiom 3 says that $\forall x\,[\alpha \to \beta] \to \forall x\,[\alpha] \to \forall x\,[\beta]$. Now consider any structure $\mathfrak{A}$ and value-function s. If $\mathfrak{A}, s \models \forall x\,[\alpha \to \beta]$ then for all $a \in A$ (where A is the domain of $\mathfrak{A}$), the implication $\alpha_a^x \to \beta_a^x$ is true. Furthermore, if $\mathfrak{A}, s \models \forall x\,[\alpha]$ then for all $a \in A$ the statement $\mathfrak{A}, s \models \alpha_a^x$ is true. The so verbally described connections are then formalised in Axiom 3.

Axiom 4 says that whenever x does not occur free in α then the formula $\alpha \to \forall x\,[\alpha]$ is valid. This is indeed the case, because when the truth of α does not depend on x and $\mathfrak{A}, s \models \alpha$, then also $\mathfrak{A}, s(x|a) \models \alpha$ for all $a \in A$. So it follows that $\mathfrak{A}, s \models \forall x\,[\alpha]$. For that reason, $\mathfrak{A}, s \models \alpha \to \forall x\,[\alpha]$, independently of whether $\mathfrak{A}, s$ make α true or not. So the formulas of Axiom 4 are also all valid.

Axiom 5 says that $x = x$ is valid for all x. This statement is true, indeed in general even $t = t$ for all terms t, as for every structure $\mathfrak{A}$ and every function s, $\overline{s}(t)$ is on both sides of the equality sign the same value.

Axiom 6 says that if $x = y$ and α, β are atomic, thus quantifier-free, formulas in which some occurrences of x and y are interchanged, then this has no effect, as in all terms where x is replaced by y or vice versa, the values $\overline{s}(x)$ and $\overline{s}(y)$ are the same by the assumption that $\mathfrak{A}, s \models x = y$. It follows that whenever $\mathfrak{A}, s \models x = y$ then $\mathfrak{A}, s \models \alpha$ iff $\mathfrak{A}, s \models \beta$. Thus $\mathfrak{A}, s \models x = y \to \alpha \to \beta$ and also this formula is valid.

Sountness Theorem. If $S \vdash \alpha$ then $S \models \alpha$.

Proof. Let $\beta_1, \beta_2, \ldots, \beta_n$ with $\beta_n = \alpha$ be a proof for $S \vdash \alpha$. For $m = 1, 2, \ldots, n$, one can show by induction that $S \models \beta_m$, using the following case distinction.

1. $\beta_m \in S$: Clearly $S \models \beta_m$.

2. $\beta_m \in \Lambda$: Now β_m is valid and therefore $S \models \beta_m$.

3. β_m is obtained from β_i, β_j with $i, j < m$ by Modus Ponens, that is, $\beta_j = \beta_i \to \beta_m$: By induction hypothesis, $S \models \beta_i$ and $S \models \beta_i \to \beta_m$; in other words, whenever $\mathfrak{A}, s$ makes all formulas in S true, then $\mathfrak{A}, s$ makes also β_i and $\beta_i \to \beta_m$ true. However, these $\mathfrak{A}, s$ then make also β_m true; so $S \models \beta_m$.

Thus $S \models \beta_n$ and by $\beta_n = \alpha$, it follows that $S \models \alpha$. $\square$

Completeness Theorem versus Satisfiability Theorem. Recall that a set S of formulas is satisfiable iff there is a structure $\mathfrak{A}$ and a function s from the variables to the domain of $\mathfrak{A}$ such that $\mathfrak{A}, s \models S$. Now the following two statements are equivalent:

`(a)` For all sets S of wff and all wff α, if $S \models \alpha$ then $S \vdash \alpha$;

`(b)` For all sets S of wff, if S is consistent then S is satisfiable.

For the equivalence, assume that `(a)` holds and let S be any set of formulas. Let S be any set of formulas which is consistent. If now S is not satisfiable, then $S \models \alpha$ for a antitautology α. Now by `(a)` it follows that $S \vdash \alpha$ and S is not consistent, in contrary to the assumption. Thus S must be satisfiable.

Now assume that `(b)` holds and consider any S and α with $S \models \alpha$ and $S \cup \{\alpha\}$ only consisting of wff. If now $S \vdash \alpha$ would not hold then $S \cup \{\neg\alpha\}$ would be consistent and by `(b)` there are a structure $\mathfrak{A}$ and a value-function s with $\mathfrak{A}, s \models S \cup \{\neg\alpha\}$. But as $S \models \alpha$, such a $\mathfrak{A}, s$ cannot exist; hence $S \cup \{\neg\alpha\}$ is inconsistent and one can do the following derivations:

1. $S \cup \{\neg\alpha\} \vdash \alpha$ (Due to inconsistency);

2. $S \vdash \neg\alpha \to \alpha$ (Deduction Theorem);

3. $S \vdash (\neg\alpha \to \alpha) \to \alpha$ (Axiom 1);

4. $S \vdash \alpha$ (Modus Ponens).

Thus `(a)` holds.

Gödel's Completeness Theorem. If $S \models \alpha$ then $S \vdash \alpha$.

Proof. This theorem is proven by showing the following equivalent statement: Every consistent set of formulas is satisfiable.

Let S be a consistent set of wffs in first-order logic. Then the main goal is to extend S to a set T such that T satisfies the following items:

(I) $S \subseteq T$;

(II) T is consistent and satisfies for all formulas α, that either $\alpha \in T$ or $\neg\alpha \in T$;

(III) For every variable x and every formula α there is a new constant $c_{x,\alpha}$ such that

$$\exists x\,[\alpha] \rightarrow \alpha^x_{c_{x,\alpha}}$$

is in T; here $\exists x\,[\alpha]$ stands for $\neg\forall\,[\neg\alpha]$.

After that, in Step 4, one forms a structure $\mathfrak{A}$ in which all members of S not containing $=$ are satisfied and in Step 5 and 6 one adjusts the structure such that also $=$ is handled properly. Now an outline of the proof.

Step 1 One expands the language by adding countably many new constant symbols. S remains consistent with respect to this enlarged language.

Step 2 One selects for each formula α and each variable x in the new language constant $c_{\alpha,x}$ and adds the formula

$$\exists x\,[\alpha] \rightarrow \alpha^x_{c_{x,\alpha}}$$

to S, just giving S'; here the constant $c_{x,\alpha}$ does not appear in α and one can process the formulas in a sequence $\alpha_0, \alpha_1, \ldots$ such that when c_{x_m,α_n} is chosen then this constant does not appear in any of the formulas α_k with $k \leq n$ and it also is different from all constants taken for formulas with pairs $(x_{m'}, \alpha_{n'})$ processed before (x_n, α_n).

Step 3 Now one inductvively extends S' to a set $T = \bigcup_s T_s$ with $T_0 = \emptyset$ by doing the following for each formula $\alpha_0, \alpha_1, \ldots$: If $T_s \cup \{\alpha_s\}$ is consistent then $T_{s+1} = T_s \cup \{\alpha_s\}$ else $T_{s+1} = T_s \cup \{\neg\alpha_s\}$.

Step 4 Now one defines a structure $\mathfrak{A}$ as follows: First one introduces a new predicate E and then considers instead of T all formulas in a set T' obtained by replacing in all primitive subformulas of the form $t_1 = t_2$ by $E(t_1, t_2)$. Now one lets the

domain A of $\mathfrak{A}$ be the set of all terms using the enlarged set of constants and one defines that a predicate $P(t_1, \ldots, t_n)$ is true in $\mathfrak{A}$ iff $P(t_1, \ldots, t_n) \in T'$ (note that $E(t_1, t_2) \in T'$ iff $t_1 = t_2 \in T$). For each function symbol f, one defines that in $\mathfrak{A}$, the value of f on inputs $t_1, \ldots, t_n$ is just the term $f(t_1, \ldots, t_n)$. For each variable x, one defines $s(x)$ to be some constant c such that $E(x, c) \in T'$, this constant c always exists. Furthermore, for the variable assignment s, one defines $\overline{s}(t) = t$ and says that $\mathfrak{A}, s \models \alpha$ iff $\alpha \in T'$.

Step 5 In the next step, one defines a further structure $\mathfrak{B}$ by forming the equivalence classes of E on A, that is, for every term h, one lets $h(t) = \{t' \in A : E(t, t') \in T'\}$ and $s'(x) = h(s(x))$ and $B = \{h(t) : t \in A\}$. Now one observes that $h(f(t_1, \ldots, t_n)) = f(h(t_1), \ldots, h(t_n))$ for all function symbols f and that h is a homomorphism from A to B. One defines that $P^{\mathfrak{B}}(h(t_1), \ldots, h(t_n))$ is true iff $P^{\mathfrak{A}}(t_1, \ldots, t_n)$ is true and that $t_1 = t_2$ is true in $\mathfrak{B}$ iff $E^{\mathfrak{A}}(t_1, t_2)$ is true. One can prove by structural induction that $\mathfrak{B}, s' \models \alpha$ iff $\alpha \in T$.

Step 6 For getting the final structure $\mathfrak{C}$, one just defines $\mathfrak{C}$ to have the same value as $\mathfrak{B}$ on all symbols which were originally in the logical language and one keeps s'. Then one sees that $\mathfrak{C}, s' \models \alpha$ for all $\alpha \in S$.