Interreg
EUROPEAN UNION
Austria-Czech Republic
European Regional Development Fund

INFORMATICS

Computer science

UNIVERSITY OF APPLIED SCIENCES UPPER AUSTRIA

VŠTE

EUROPEAN UNION

Contents

1. BASIC MATHEMATICAL CONCEPTS AND NUMERICAL SYSTEMS

- Data or information we can capture with our senses.
- Information is data, which we understand, make sense and can be quantified (measured) after conversion into numbers.
- Information can be: Text, Image, or Sound.
- Informatics is a science of preservation, processing by using data and information. It is Part I of mathematics and as a means to use computer technology.
- IT branches:
    - Computing Technology - Examines hardware
    - Algorithmization - design of problem solving procedures
    - Programming - converting algorithms into programming language
    - Software Engineering - Application Development
    - Computer Graphics - Learn about creating and processing 2D and 3D images
    - Computer modeling and simulation - Application of mathematical models to real-world situations
    - Formal logic, automata theory and formal languages - mathematical models of machines and formal languages for writing algorithms and programs
    - Cybernetics and robotics - development of machines capable of independent activities
    - Artificial Intelligence - Exploring the processes of human thinking, their mathematical description and application in machine development
- Hardware or computer hardware, a generic name for all the physical devices your computer is equipped with
- Software - It can be divided into a system and application.
- Information stored in we are measuring computers in Bit - b - and Byte - B units
- The information stored on your computer is measured in Bit - b - and Byte - B
- The bit is the base and the smallest unit of information has two values: 0, 1
- Byte as a unit of information has an 8-bit value.

Since informatics is part of mathematics, it also uses some of its terms, such as:

- Cartesian product
- Relation
- Morphism
- Divisibility
- The largest common divisor, the smallest common multiple
- Prime Numbers

Because the computer stores information in bits and multiples, it works in a binary numeric system. You need to say something about numerical systems.

Numerical systems are divided into positional and non-positional.

Position Numerical Systems - The value of each digit is given by its position in the symbol sequence

Their advantage is high elasticity and a relatively small set of digits; the disadvantage is a very easy change of the value of a number by simply adding a digit to the original number.

The key characteristic is the Base, then the weights of the individual digits are the basis of the base.

Examples:

Unary – the base 1

Binary - the base 2, two states - yes / no, 1 bit, the symbols 0, 1,

Octal - the base 8, one byte (= 8 bits), the symbols 0, 1, 2, 3, 4, 5, 6, 7

Decimal - the base 10, natural for people (ten fingers), the symbols 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 (the number)

Hexadecimal – the base: 16, 2 bytes (= 16 bits), symbols: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

Sexagesimal – the base 60, time measurement

Non-Position Numeric Systems - A method of representing numbers in which the value of a digit is not given by its location in a given sequence of digits. These ways of writing numbers are hardly used today and are considered obsolete, yet they have certain advantages such as simple addition and subtraction. The drawbacks are that they often did not contain a symbol for zero and negative numbers, as well as a long count of numbers that significantly exceed the value of the largest symbol of the system

Example: Roman numerals

Transfer between systems

From a higher base system

Converting from a higher base system to a lower base system then we usually use a partitioning algorithm.

73/2 = 36 + 1 - 1 at the 1st place from behind
36/2 = 18 + 0 - 0 at 2nd place from behind
18/2 = 9 + 0 - 0 at 3rd place from behind
9/2 = 4 +1 - 1 at the 4th position from behind
4/2 = 2 + 0 - 0 at the 5th place from behind

2/2 = 1 + 0 - 0 to the 6th position from behind
1/2 = 0 +1 - 1 at the 7th position from behind

From a lower base system

Transferring from a lower base system to a higher base system is easier. We know that the digit at the n-th place (calculated from 0) represents the number resulting from the amplification of the base on the n-th. These numbers are numbered and transferred.

$$1 \cdot 2^0 + 0 \cdot 2^1 + 0 \cdot 2^2 + 1 \cdot 2^3 + 0 \cdot 2^4 + 0 \cdot 2^5 + 1 \cdot 2^6 = 73_{10}$$

2. LOGIC

Logic is the science of correct reasoning, or the art of proper argumentation.

What is the judgment (argument)?

Based on the truth of the premise (assumptions) P1 ... Pn it can be concluded that the conclusion Z is true

We recognize several types of argumentation logic.

Deduction

Conclusion Z logically follows from the assumptions $1,\ldots, Pn$, we denote $P1,\ldots, Pn \mid = Z$, if under no circumstances can the case arise such that the assumptions would be true and the conclusion would be untrue.

Example:

P1: All rabbits out of the hat are white.

P2: These rabbits are out of the hat

Z: ⇒ These rabbits are white.

Induction

Generalization - from specific to general

Example:

P1: These rabbits are out of the hat

P2: These rabbits are white.

Z: ⇒ (probably) All rabbits in the hat are white.

Conclusion Z only applies with a certain probability

Abduction

We create hypotheses for observed phenomena, diagnosis of "disorders"

Example:

P1: All rabbits out of the hat are white.

P2: These rabbits are white.

Z: ⇒ (probably) These rabbits are out of the hat

Conclusion Z only applies with a certain probability

Examples:

P1: All green toadstools are violent poisonous

P2: This fungus is a green toadstool.

Z: This fungus is violently poisonous.

Valid deduction

P1: All green toadstools are violently poisonous

P2: This pencil is green a green toadstool.

Z: This pencil is violently poisonous.

Correct Judgment, Conclusion False ⇒ At least one premise is untrue

Judgment has the correct logical form

Logical connectives - "and", "or", "if", "then", and others have fixed meanings, interpret them using elementary statements or parts thereof (predicates and functional terms).

Logic is a tool that helps to discover the relationship of logical outcome, to solve tasks such as "What does this imply?" It helps our intuition, which can sometimes fail, because premise can be complexly formulated, intertwined and negated, the relationship of occurrence is not obvious at first view.

Example:

P1: All men like football and beer

P2: Some beer lovers do not like football

P3: Xaver likes only lovers of football and beer

Z: Xaver does not like some women.

Essentially, if all assumptions are true, then the conclusion must be true.

Is Judgment Valid?

> Of course, if Xaver likes only lovers of football and beer (3.), he does not like some beer lovers (those who do not like football (2.)), he does not like (according to 1.) some "non-men" i.e. women.

However, it is not valid under the definition

Judgment is valid if necessary, i.e. under all circumstances (interpretations) when true assumptions are true and true.

But: in our example, those individuals who are not men would not have to be interpreted as women. There is no premise here, "who is not a man, is a woman," likewise we still need the premise "who is a lover of something he likes".

We must put all the assumptions necessary to draw the conclusion

P1: All men like football and beer.

P2: Some beer lovers do not like football.

P3: Xaver likes only lovers of football and beer.

P4: Who is not a man is a woman.

P5: Who's a lover of something, he likes it.

Z: Some women do not like Xaver.

Now the judgment is correct, it has a valid logical form.

Conclusion logically follows from the assumptions (they are "informally, deductively included").

Logic properties

> Valid (correct) judgment may have a false conclusion
>
> > Usage: Proof AD ABSURDUM
>
> Monotony
>
> > If judgment is valid, extending the set of assumptions to a further assumption does not lead to a change in the validity of the judgment
>
> From the controversial assumptions, any conclusion is drawn
>
> Reflexivity, Transition

Valid Judgment Schemes

1. $A \supset B, A| = B$

   modus ponens

   P1: No prime number divisible 3

   P2: 9 is divisible 3

   Z: 9 is not a prime number

2. $A \supset \neg B, B| = \neg A$

   modus ponens + transposition

   P1: All people are reasonable

   P2: The stone is not reasonable

   Z: This stone is not human

3. $A \supset B, \neg B| = \neg A$

   modus ponens + transposition

   P1: If 12 is prime or not divisible 3

   P2: 12 is divisible 3

   Z: 12 is not a prime number

4. $\neg A \vee \neg B, B| = \neg A$

   elimination of disjunction

   P1: 12 is not a prime number or is not divisible 3

   P2: 12 is divisible 3

   Z: 12 is not a prime number

3. SETS AND RELATIONS

3.1. Sets

Definition - a summary of objects that are precisely defined and identifiable and forms part of the world our ideas and thoughts; these objects are called elements of the set.

We use the following symbol for sets:

- Sets: A, B, C ...
- Elements : a, b, c ...
- Element Sets : $a \in A$
- For all x applies P : ∀x : P
    - There is x for which P holds: ∃x : P
    - There is just one x such that P holds : ∃!x : P
- Conjunctions, disjunctions, negations: ∧, ∨, ¬
- Implications, Equivalences: ⇒ , ⇔
- Sum, multiplication: Σ, Π

The set can be made

- Enumeration of elements : $A = \{1,2,3,4,5\}$
- Using the characteristic features: $A = \{a \in N | a < 6\}$

We define empty set: $\{\emptyset\}$, which contains no elements.

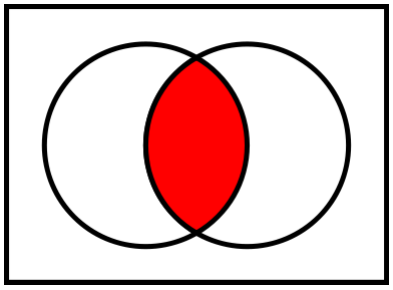
The number of element elements determines - **cardinality** - | A |, where A is a set.

We recognize sets:

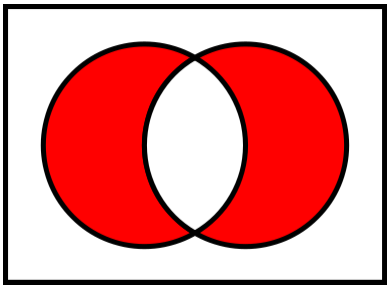
- Finite
- Infinite
- countable
- Continuum

Basic Operations with sets:

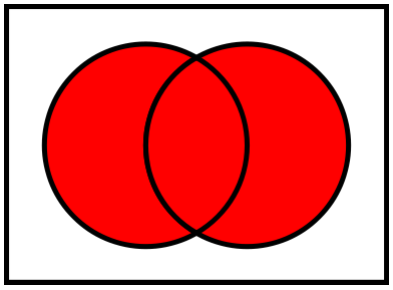
Intersection: $A \cap B$



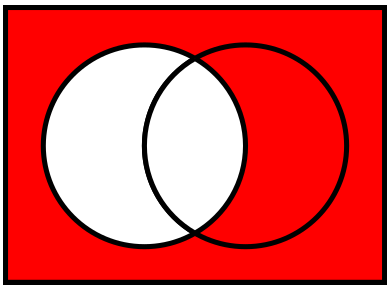
Symmetrical difference: $A \triangle B$



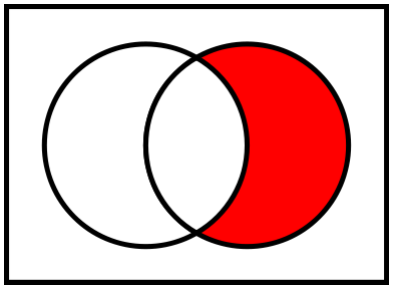
Union: $A \cup B$



Complement of A in U: $A^c = U \setminus A$



Difference of sets: $B \setminus A$



Subset (inclusion– the opposite of exclusion): $A \subseteq B$

$$(\forall x)(x \in A \Rightarrow x \in B)$$

Potential set - set of all subsets:

$$P(\emptyset) = \emptyset$$

$$P(\{a\}) = \{\emptyset, \{a\}\}$$

$$P(\{a, b\}) = \{\emptyset, \{a\}, \{b\}, \{a, b\}\}$$

3.2. Relations

Definition: n-ary relation between sets $A_1, A_2, A_3, \ldots, A_n$, where $n \in N$, we mean any subset of the Cartesian product *of n* sets.

Cartesian product (discrete product)

Set $X \times Y$, which contains all the ordered pairs, where the first entry is from X and the second entry is z Y

A\B	1	2	3
X	(X,1)	(X,2)	(X,3)
Y	(Y,1)	(Y,2)	(Y,3)
Z	(Z,1)	(Z,2)	(Z,3)

Relations can be classified by arity to:

Unary - each subset of the set *N*

Example: a positive number is a true / false statement

Binary - Each set of ordered pairs $[x; y] \in M^2$

Eg: is greater than, is smaller than, is a subset

Ternary - Each set of ordered triplets $[x; y; z] \in M^3$.

Example: lies between

N- aray - each set of ordered n-tuplets of M^n.

Operations with relations

In addition to classical set operations (all sessions must have the same arithmetic), we introduce the inverse relation as: R^{-1}: $\forall a \in M_1 \forall b \in M_2$:[a,b] $\in R \Leftrightarrow$ [b,a] $\in R^{-1}$ and a compound session.

The binary relation on the set is:

- **reflexive**, if for all a ∈ M holds that [a, a] ∈ R.
- **Symmetrical** if for all a, b ∈ M applies if there is [a, b] ∈ R then [b, a] ∈ R.
- **Antisymmetric** if for all a, b ∈ M holds that if it is [a, b] ∈ R and also [b, a] ∈ R, then a = b.
- **Transitive** if for all a, b, c ∈ M holds that if [a, b] ∈ R while [b, c] ∈ R then [a, c] is in R.

Two special relations

- **Equivalence** is a relation that is reflexive, symmetric and transitive.
- **Order** is a relation that is reflexive, antisymmetric and transitive.

4. RELATIONAL STRUCTURES

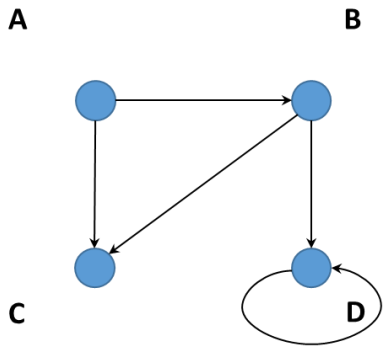
Relational structure means a mathematical structure that, in addition to the carrier set, contains one or more relations (which may have different and different arithmetic). These include oriented graphs and ordered sets (including their special cases, lines arranged linearly or well).

Directed graphs

A directed graph G is a pair (V, E), where E is a subset of the Cartesian product V × V × V $E \subseteq V$

Elements E are called arrows or oriented edges. Oriented edge e has the form (x, y). We say that this oriented edge is based on x and ends in y.

Directed graphs are represented by adjacency matrix, or list of vertices and edges



Adjacent Matrix :

		WHERE			
		A	**B**	**C**	**D**
FROM	**A**	0	1	1	0
	B	0	0	1	1
	C	0	0	0	0
	D	0	0	0	1

For directed graphs may not be symmetric, 1where is the edge, 0 where edge is missing

The set of vertices and edges G: (A, [A, B]), (A, [A, C]) (B, [B, C] D, D])

Ordered sets

Linear arrangement (= total ordering) is defined so that:

Each two elements of the linearly ordered set are comparable, ie there is a relation R on the set ***X*** and *a, b, c* ∈ ***X*** such that:

Transitivity: $(\forall x, y, z \in A)\big((xRy \wedge yRz) \Rightarrow xRz\big)$

Weak antisymmetry : $(\forall x, y \in A)\big((xRy \wedge yRx) \Rightarrow x = y\big)$

Trichotomy: $aRb \vee bRa \vee a = b$

Example: arrangement on a set of natural and real numbers

A well-arrangment is defined as:

The set S is well-arranged just when each non-empty part of the ordered set S has the smallest element

Example: Natural numbers

An ordered set is one on which an arrangement is defined.

Arrangement is a binary relation *R*, which is reflexive, transitive and weakly antisymmetric

$(\forall x \in A)(xRx)$ reflexivity - each element is in a *R* - relation to itself

$(\forall x, y, z \in A)\big((xRy \wedge yRz) \Rightarrow xRz\big)$ transitivity - if the element of the set is in the arrangement between two other elements, the two are also comparable

$(\forall x, y \in A)\big((xRy \wedge yRx) \Rightarrow x = y\big)$ weak antisymmetry (no cycles in arrangement)

= **fuzzy arrangement**

Sharp Arrangement - Reflexivity replaced by antireflexivity - no element is in relation with itself $(\forall x \in A)\big(\neg(xRx)\big)$

5.PROJECTION

A special example of a binary relation that ensures unambiguous image to each pattern

If this is a binary session on a numeric set, then we call it as a function.

Definition:

Morphism f from the set ***A*** to the set ***B*** is such a binary relation for which each element x of ***A*** assigns no more than one element y of ***B*** so that $[x, y] \in f$

Designation:

$y = f(x) \quad f: x \to y$

Important notation: $y = f(x) \in B,\ x \in A$

y - The target of element x in morphism f , morphism value f at point x

x - Source of element y in view f

A set of elements $x \in A$, for which there is just one such element $y \in B$, that $y = f(x)$, is called the domain of f

B set of elements $y \in B$, for which there is at least one such element $x \in A$, that $f(x) = y$ is called the range of f

We distinguish several basic types of morphisms:

Morphism in Set (Morphism on Set):

$$f: A \to B,\ A = B$$

Set into a set - the whole set *A* is domain

$$f: A \to B,\ \ D(f) = A$$

From set to set (surjection) - the whole set *B* is a range of values

$$f: A \to B,\ \ H(f) = B, \forall b \in B: \exists a \in A: f(a) = b$$

Set on set

$$f: A \to B,\ \ D(f) = A \wedge H(f) = B,$$

$$\forall a \in A: \exists! b \in B: b = f(a) \wedge \forall b' \in B: \exists! a' \in A: f(a') = b'$$

Injection

$$\forall b \in B: \exists! a \in A: f(a) = b$$

Unambiguous (bijection) – Injection function of set ***A*** on set ***B*** (is injectable and surjective at the same time)

$$f: A \leftrightarrow B$$

Inverse - There are defined only for injection functions

If it is $f: A \to B$ injection, then morphism

$f^{-1}: B \to A$, which $\forall b \in H(f)$: $f^{-1}(b) = a \in D(f): y = f(x)$, is called inverse

$$D(f^{-1}) = H(f),\ f^{-1}(y) = x \Leftrightarrow f(x) = y$$

Morphisms can also be composed:

Sets ***A*** , ***B*** , ***C***

Morphism: $f: A \to B$ and $g: B \to C$

Composed morphism: $h: A \to C,\ h = g \circ f,\ h(a) = g\big(f(a)\big)\ \forall a \in A$

It is generally not commutative, but is associative

Inversion of a composed morphism:

$$(g \circ f)^{-1} = f^{-1} \circ g^{-1}$$

6.BOOLEAN ALGEBRA

Boolean algebra is defined as: Sixth $(A, \wedge, \vee, -, 0, 1)$, where *A* is non-empty set, $0 \in A$ - the smallest element , $1 \in A$ - the largest element , $-$ unary operation (complement), ∧ -intersection, ∧ logical product , ∨ - logical sum .

It is based on axioms:

Commutativity

$$x \vee y = y \vee x$$

$$x \wedge y = y \wedge x$$

Distributivity

$$x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$$

$$x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$$

Neutrality 0 and 1

$$x \vee 0 = x$$

$$x \wedge 1 = x$$

Complementarity

$$x \vee -x = 1$$

$$x \wedge -x = 0$$

A has the following characteristics:

- Absorption: $x \vee (x \wedge y) = x$, $x \wedge (x \vee y) = x$
- Aggression of zeros: $x \wedge 0 = 0$
- Aggression of one: $x \vee 1 = 1$
- idempotence : $x \vee x = x$, $x \wedge x = x$
- Absorption negation: $x \vee (-x \wedge y) = x \vee y$, $x \wedge (-x \vee y) = x \wedge y$
- Double negation: $-(-x) = x$
- De Morgan's laws: $-x \wedge -y = -(x \vee y)$, $-x \vee -y = -(x \wedge y)$
- *0* and *1* are complementary to one another: -0 = 1, -1 = 0

Basic one-input operations:

- ID (Identity)
- NOT (negate)

Two Input Basic Operations.

- OR (logical sum)
- AND (logical product)

Truth table for basic operations:

A	B	ID(A)	ID(B)	NOT(A)	NOT(B)	A OR B	A AND B
0	0	0	0	1	1	0	0
0	1	0	1	1	0	1	0
1	0	1	0	0	1	1	0
1	1	1	1	0	0	1	1

Derived two-input operations:

- NOR (Not OR – negation of sum)
- NAND (Not AND - negating the product)
- Implication ⇒ (if the assumption A is fulfilled, B results, if the expectation is not fulfilled, something results and therefore 1)
- Equivalence of EQ ⇔ (matching of inputs)
- XOR (Exclusive OR - Unique Inputs)

Truth table for derived two-input operations:

A	B	NOR	NAND	⇒	⇔	XOR
0	**0**	1	1	1	1	0
0	**1**	0	1	1	0	1
1	**0**	0	1	0	0	1
1	**1**	0	0	1	1	0

Let's take the expression: A (B + NOT (C))

You can write this expression using:

1. **Truth tables:**

i	A	B	C	A(B+NOT(C))
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	0
4	1	0	0	1
5	1	0	1	0
6	1	1	0	1
7	1	1	1	1

2. **Algebraic expression:**

AND (NOT (B) (NOT (C)) + AB (NOT (C)) + ABC

This is the sum of all expressions giving the result 1

3. **Set the state index so that the lines in the truth table are numbered 0** (see Column *i*)

A (B+NOT(C)) = {4, 6, 7}

To minimize the algebraic expression, Karnaugh's map can be used.

It is filled in so that the columns (rows) above which the variable is given are for it 1

The table shows the values in the ABC order

			B	
		C		
	000	001	011	010
A	100	101	111	110

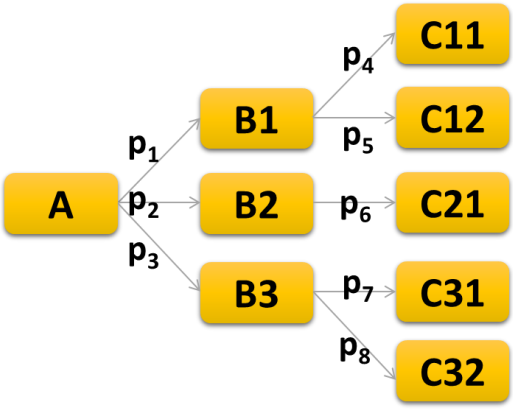
Example : A (NOT (B) (NOT (C)) + A (NOT (B)) C + ABC = A

			B	
		C		
	0	0	0	0
A	1	0	1	1

7.PROBABILITY AND STATISTICS

The difference between determinism and stochasticity is in that determinism is known in advance. FROM A to B goes with 100% probability (I'm always going to do that , another option is not).

In contrast, in the stochastic phenomena everything is just happening with with some probability, from And to B1 I'm going with probability p_1. But I can also go to B2 with probability p_2 and B3 s p_3. That is, I only know the probability with what happens the result. IN Random processes play a role in this phenomenon. If we look at the image, the sum of p1, p2 and p3 must be equal to 1, or z And I'm going to some B with 100% probability. Similarly, the probability p6 (B2-> C21) is equal to one .



Probability is defined as follows:

Let F be the experiment E, the set of conditions of the experiment Π and the set of all possible results F. F contains events 0, A, B... Ω, where 0 is an event that never occurs and Ω event that always occurs. Therefore, the experiment can be written as E = (Ω, F). The set of experiment results repeated under the same conditions is N, the individual results are 1 ... n and I is a subset of the results. Subset *A(I)* is subset of *I* such that the result of the experiment is an event *A.* The number of results in the subset *A (I)* is $n_A(I)$ and the set of all results is *Ω (I)* . Then all subsets *I* apply , where *P (A)* is the probability of the phenomenon *A*:

$$P(A) \approx \frac{n_A(I)}{n_\Omega(I)}$$

The relative frequency of the *A* phenomenon can be defined as : $\frac{n_A(I)}{n_\Omega(I)}$

Probability properties - the probability of A is always between 0 and 1 including both extreme values.

The probability that phenomenon A does not occur, or negation of phenomenon A is 1-P (A).

If phenomena A and B do not occur simultaneously, then P $(A \cap B) = \emptyset$

Probability of occurrence of phenomenon *A* or phenomenon *B:* $P(A \cup B)$ is $P(A \cup B) = P(A) + P(B)$

Conditional probability - probability of A phenomenon when phenomenon B occurs: $(A|B) = \frac{P(A \cap B)}{P(B)}; P(B|A) = \frac{P(A \cap B)}{P(A)}$

$$\boldsymbol{(A|B) = \frac{P(A \cap B)}{P(B)}; P(B|A) = \frac{P(A \cap B)}{P(A)}}$$

Bayes theorem: $P(A|B) \cdot P(B) = P(B|A) \cdot P(A)$

Statistics is a set of concepts, rules and procedures that help us organize numerical information in the background to understand the techniques and make informed decisions.

Statistical knowledge is applied to the data, which is the facts and information from observations (ideally from experiments)

Data are divided into numerical (quantitative) data, which are the result of the measurement, and categorical (qualitative) data, divided into groups based on common characteristics.

The basic statistical variables are the mean value (weighted average), the median, which shows the central tendency, the middle value in the given set, and the modus as the most common value. Next, we calculate the variance, and from it the standard deviation as the root. We can also find covariance that determines how much the two values change together. And then skewness and kurtosis, revealing the distribution of values for a given file.

The distribution of file values describes the so-called distribution. One of the basic distributions is Gaussian (normal) distribution. It is symmetrical, having the same mean, mode, median, skewness and kurtosis are zero.

One of the basic statistics is the Central Limit Theorem, which expresses the fact that the sum of many independent random variables will tend to disperse in a small set of distribution functions.

8. THEORY OF INFORMATION

Information theory deals with the measurement, transmission, encryption, storage and subsequent processing of information in quantitative terms. Its founder is C.E. Shannon.

Information

What we exchange with the outside world when we adapt to it and adapt to it by adapting it

Content of the message, communication, clarification, explanation, instruction.

Data, numbers, characters, commands, instructions, commands, messages and the like. For information, consider also the ideas and perceptions received and emitted by living organisms.

The size information is provided orderliness (uncertainty) of the elements (system).

The system information is the greater the likelihood of occurrence of individual states is smaller. The information is bigger if the message contains something new that was not known or could not be easily guessed (it is unlikely)

For passing the information is needed

Any method of encoding = transfer to appropriate signals or symbols

Signal - information-bearing physical quantity

Message - Expression of information using a sequence of symbols (characters)

The report as such has three parts:

- Syntax - track
    - Syntactic content
    - It can exist itself without semantic and pragmatic content
    - It refers to the mutual arrangement of characters as carriers of information
    - Describes the quantitative information page
- Semantics - Information Content
    - Semantic content
    - The meaning of the message can not be measured
    - A message with the same amount of information can be written in different languages
    - Describes qualitative information page
- Importance - pragmatic content
    - Determines the significance (usefulness, value) of the message and priority
    - Qualitative information page

The information content of the message is measured by entropy.

Information is a measure of the amount of uncertainty or uncertainty about any random action eliminated by the realization of this happening.

We determine the degree of uncertainty as the entropy of the system

$$H(X) = -\sum_{i=1}^{n} p(x_i)p(x_i)$$

Information about System X can be obtained from:

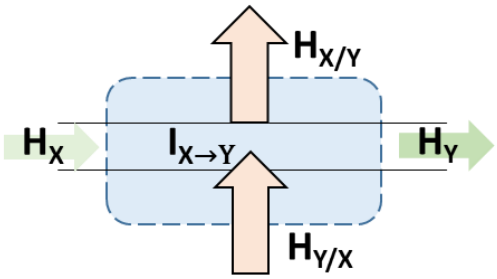
Direct observation

Indirectly through a system of Y (X system is unavailable to us)

System Status Y is not necessarily identical with the state of X (the text of the telegram in one city X, Y in the second city)

Differences of two kinds

- Some states of the X system appear in one state Y (Y can not distinguish the fines in X, Y is coarser than X)
- Errors in transmission between X and Y - eg: noise
- Using a channel



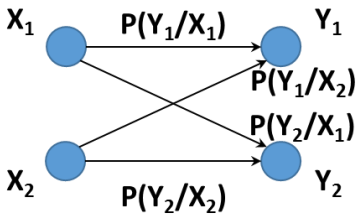
$H_{X/Y}$ - average lost information

$H_{Y/X}$ - average disturbing information

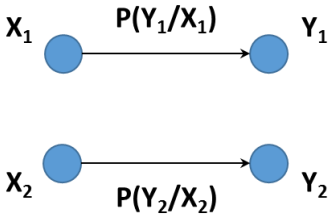
$I_{X \to Y}$ - mutual information

For channel information, we distinguish two types of channels:

- **Noisy:**



- **Noiseless:**



P (Y_1 / X_1)	probability when sending element X_1 Y_1 received
P (Y_2 / X_2)	probability when sending element X_2 Y_2 receive
P (Y_1 / X_2)	probability when sending element $_2$ receive X Y_1
P (Y_2 / X_1)	the probability that we get Y_2 when sending element X_1

Channel permeability (channel capacity) - Channel ability to transmit information. Maximum information that can be transferred per unit of time.

$C = B\left(1 + \frac{S}{N}\right)$, where *B* is the channel width, *S* is the signal, *N* is noise.

Sampling theorem

An accurate reconstruction of a continuous, frequency restricted signal from its samples is possible if the sampling rate is greater than twice the highest harmonic component of the sampled signal.

The minimum possible length of signal elements

$\tau_0 = \frac{1}{2F_m}$, F_m is the limit frequency, bandwidth

9. INFORMATION THEORY APPLICATION

For example, Huffman encoding, arithmetic coding, LZW, JPEG, MP3, TIFF, error detection and error correction codes, statistical applications, and MDL are included in the theory of information.

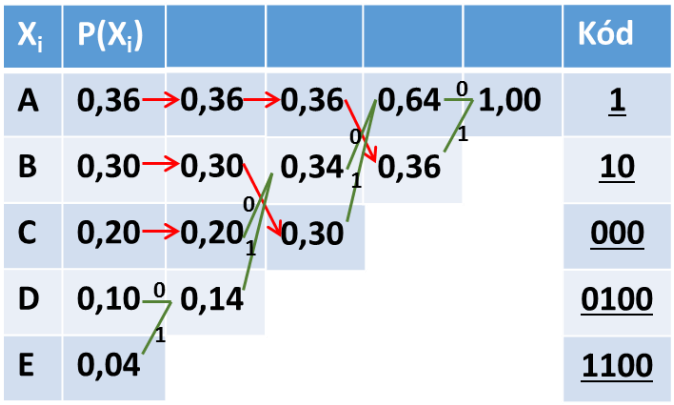
Huffman encoding is an algorithm for lossless data compression. Converts the characters input file into bit strings of different lengths. The most frequent symbols into bit strings with the shortest length (even 1bit). Least frequent to longer chains (can be longer than 8 bit).

It has two phases

- Passes the file and creates statistics
- Creates a binary tree to compress data

The code is made from the leaves to the root.

Example:



Shannon-Fano coding is the statistical method of lossless compression. Huffman's encoding differs only in the binary tree design. The character set is recursively divided into two roughly equal subsets. One sub-code is then assigned a binary 1 and the second 0. The code is thus constructed from the root to the leaves, unlike Huffman coding, the code is formed from the leaves to the root, may not be optimal.

Example:

Char	p(x)	s	Groups				Code
A	0,36	1	0				0
B	0,3	0,64	1	0			10
C	0,2	0,34		1	0		110
D	0,1	0,14			1	0	1110
E	0,04	0,04				1	1111

Arithmetic coding is a batch-length compression of variable length data of a code word. It encodes the entire text into a single number, a fraction $n \in < 0; 1)$. It is in the propeller difficult to count with real numbers. For a longer report, we would no longer be able to achieve the necessary precision, and some subintervals might begin to fuse us. It uses block compression, where the blocks are large enough to ensure sufficient accuracy when distributing.

Information can be secured during transmission to detect and correct errors and to prevent unauthorized reading.

Codes for error detection and correction

Parity - Data For each segment is connected to another bit, whose value of the number of binary ones complement of the number is odd or even (odd / even parity)

Even: 10011011 10010101 → 100110111 100101010

Odd: 10011011 10010101 → 100110110 100101011

CRC - checksum - The data is divided into sections of the required length (8, 16, 32 bits) and these segments are added after the bits without transfer. The resulting data segment connects to data transferred.

00001101	Data
00010000	
01000100	
10010000	
11110001	**CRC**

Hamming code is a linear code used in telecommunications to detect up to two erroneous bits or to repair one bad bit

Algorithm:

All bit positions whose number is equal to Power 2 are used for the parity bit (1, 2, 4, 8, 16, 32, ...).

All other bit positions belong to the encoded information word (3, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15, 17, ...).

Each parity bit is calculated from some of the information word bits. The position of the parity bit indicates the sequence of bits that are scanned in the code word and which are skipped.

For parity bit p_1 (position 1), in the remainder of the codeword, 1 bit is skipped, 1 checks, 1 skips, 1 checks,

For parity bit p_2 (position 2) the first bit is skipped, 2 checks, 2 skips, 2 checks, etc.

For p_3 (position 4), the first 3 bits are skipped, 4 are checked, 4 are skipped, 4 is checked,

Against unauthorized reading

- Encryption – Cryptography, Historical
- Stenography - Hiding the Message
- Substitution Cipher - replacing each character with another by some rule
- Character shift - Caesar's cipher - each letter of the alphabet shifted to a fixed number of positions
- Tables of substitutions - replacing a character with another without any internal context or knowledge of the key
- Vigener Cipher - uses passwords whose characters determine the offset of the open text, so that the open text is divided into blocks of characters long as the password and each character is added with the corresponding password character
- Verman's cipher - the only known cipher that has been proven to be undetectable so far is based on the addition of open-text letters and passwords, but the password is a block of randomly selected data of the same size as the open text

Transposition grid

- Skytalé - a type of encryption that consists of a war and a wound paper or parchment on it, on which a message is written Modern
- Symmetric ciphers - a conventional cipher, uses a single key to encrypt and decrypt, DES (Data Encryption Standard), the current AES (Advanced Encryption Standard)

- Asymmetric ciphers - Use different keys (private and public key), RSA, PGP (Open-PGP) for encryption and decryption , used for electronic signature
- Hash function - a one-way mathematical function that is used, for example, to ensure data integrity.

Signing

- Electronic Signature - O of the learning of specific data that replaces a classic handwritten signature or a certified signature on the computer. It is connected to or logically connected to a data message, it enables authentication of the signed person in relation to the data message. An electronic signature is a means of verifying the identity of the sender.

10. THEORY OF COMPLEXITY

The complexity theory focuses on classifying computational problems according to their own complexity and the relationship between classes. The problem is a task that can be resolved on a computer. The problem is considered to be difficult if its solution requires considerable resources no matter what algorithm is used. Formalizes this approach, introduces computational models to study and quantify the amount of resources needed to solve problems (time, memory). Another level of complexity is a lot of communication, gate count of the circuit, the number of accesses to the cache and the number of processors. One of the goals is to determine the practical limits of what computers can calculate and what they do not.

Analysis of algorithms vs. theory of complexity vs. the theory of computability

- Algorithm analysis deals with the amount of resources needed by a particular algorithm
- The complexity theory identifies more general questions about all the algorithms that can be used to solve a particular problem. Trying to classify problems according to the limitations of available resources, introduces restrictions on available resources
- The quantification theory asks what problems can, in principle, be solved algorithmically.

The basis is a computational problem.

Problem: Infinite collection of examples and solutions to the situation, entering input, ie. An instance of the problem cannot be confused with the problem itself. The problem must be addressed regardless of its assignment.

Example: Prime Number Test – Input: Number; Output: Yes / No

You can enter the problem in several ways: The most basic way is in the form of a pronounced sentence. For solving computer is necessary translation into the language of computers. Mathematical tasks in graphs are set to for example by means of adjacent matrices.

The decision problem is the basic type of problem complexity theory, has only two outputs Yes / No. In the language where the members of the language are the cases with the answer YES and the other members with the NO answer you should be using an algorithm to decide how the specified input corresponds with this language. If YES, then the algorithm returns the input is accepted otherwise rejected. The algorithm calculates a characteristic feature of the language.

Formal languages (and problems over them) are some of the alphabet, which is not necessarily binary.

The problem function is a computational problem where one output of the total function applies to all possible inputs but is more complex than the output of the decision function. The problem of the feature is richer than the decision-making problem. It can also be redesigned to the decision-making problem as follows: we want to multiply two numbers, the answer to the decision problem is a triple (a, b, c) and the YES return is only for the triple where a * b = c

For complexity theory, you need to measure input size. The size of the input depends on the amount of time it takes to process the algorithm. These partial problems, which may be the space needed for the solution, process a separate algorithm and all relate to the size of bit entry. The complexity theory is concerned with how the algorithm will deal with the input size.

Ex.: solving connected graph of *n* edges in comparison with the graph of 2 *n* edges. If the input is *n,* then the time required to calculate the function (n) . If function $\tau(n)$ is polynomial we are talking about polynomial algorithm.

Cobham thesis - the problem can be solved in polynomial time, if it exists for the algorithm that processes the *n* -bit input in time, where *c* is a constant depending on the problem, not its input.

However, we measure not only inputs, but also the resources, whether p ro a specific algorithm or a problem.

Generally, the data size *n* , and not for a specific input value *to the* (size *n*), usually for all possible (infinite number) → size and complexity estimate usually asymptotically. IN re-source dependency is measured by:

- Elapsed time (in steps)
- Memory (in bits / bytes / cell)
- Packets (generally in frames)
- Cache (e.g. the number of accesses)

Complexity Theory class defines the complexity of the problems:

- class P
  The decision-making problem of the U lies in the class *P* if there is a Turing machine that decides the language L_U in polynomial time.
  Ex .: Finding the shortest path, minimum spanning tree
- class NP
  The decision-making role of the *U* is in the class *NP* if and only if there is a non-deterministic Turing machine that decides the language L_U in polynomial time.
  Ex .: K-color (color chart can be given up *to* colors ?) Clique of graph (exists clique in graph at least of *k* vertices?)

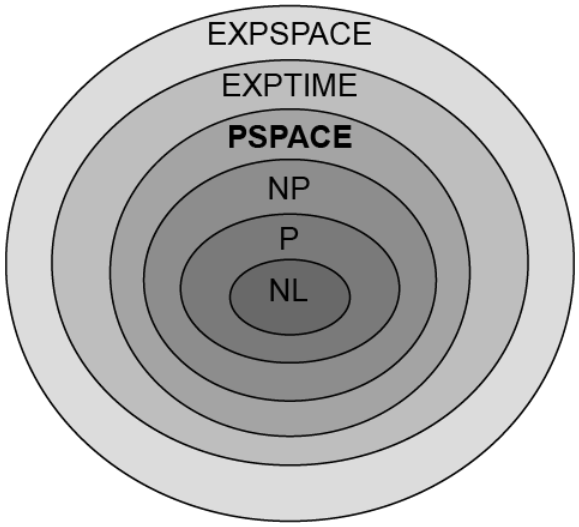
- NPC class - NP-complete problem
  The problem is in the class *NP* and also true that the polynomial reduces the role of each class *NP* . *NP* -complete tasks are the "most difficult" of all *NP* problems.
  Ex .: Traveling Salesman Problem, Knapsack Problem

Classes PSPACE and NPSPACE

The language *L* is in the class *PSPACE* if and only if there is a deterministic Turing machine that works with the memory polynomial complexity (ie. Not use any memory cell index greater than *p (n)*) and adopts the language *L* .

The language *L* is in a class *NPSPACE* just when there is non-deterministic Turing machine that works with polynomial complexity of memory and accepts the language *L* .

It has been shown that $NP \subseteq NPSPACE$ a further according *Savitch sentences* , the $PSPACE = NPSPACE$.



II. LANGUAGES AND AUTOMATA

Languages and automata are the cornerstone of theoretical computer science.

The language means any nonempty set V as alphabet and its elements are characters or symbols.

We define:

- The word above the alphabet V is the final sequence of characters from V, w = a1 a2...an
- Word length w - length of sequence w, | w | = n
- Empty word ε - sequence of length 0
- A set of all words above the alphabet V ^ *
- A set of all non-empty words above the alphabet V ^ +
- Grammar G is quadruple (N, Σ, P, S),
    - N is the final set of nonterminal symbols (nonterminals).
    - Σ is the final set of terminal symbols so that no symbol belongs to N and Σ at the same time.
    - P is the final set of derivation rules. Each rule is the form (ΣUN) ^ * N (ΣUN) ^ * → (ΣUN) ^ *
    - S is an element of N called initial symbol.

Convention

- We denote terminals - a, b, c, ...
- Chains terminals denoted - u, v, w, ...
- Individual nonterminals - A, B, C ... X, Y, Z
- Strings of nonterminals and terminals - α, β, γ, ...
- An empty string is denoted by the symbol e or even ε

Grammar is divided into several types according to the Chomsky hierarchy.

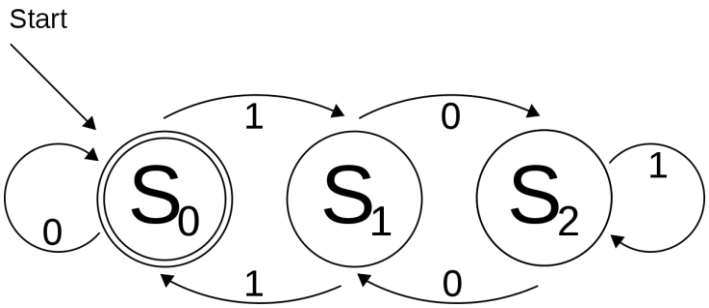
- Type 0 - All formal grammars (unrestricted grammars)
- Type 1 - Context grammars
- Type 2 - Context-free grammars
- Type 3 - Regular grammars

Finite automaton is the theoretical computational model used in computer science to study formal languages. It describes a very simple computer that can be in one of several states, between which it switches on symbols that read from the input. Set of states is finite (hence the name), finite automaton has no additional memory, in addition to the current status. We distinguish Deterministic FA - at each point in the table has just one target state - and nondeterministic FA - at every point of the table is not a target state, but the whole set of states. Also, in the transition table, there is an empty entry column, called

e. Any non-deterministic machine can be converted to deterministic. The original set of states needs to be replaced by its potential set. Each state of the machine thus created corresponds to a set of states of the original non-deterministic machine and there are clear transitions between them.

Formally, the finite automaton is defined as a ordered five (S, Σ, σ, s, A) where:

- S is a finite non-empty set of states.
- Σ is a finite non-empty set of input symbols, called alphabet.
- σ is the so called transition function (also a transition table) describing the rules of transitions between states. May either have $S \times \Sigma \rightarrow S$ (deterministic automaton) or $S \times \{\Sigma \cup \varepsilon\} \rightarrow P(S)$ (nondeterministic automaton), see below.
- s is the initial state, $s \in S$.
- A is a set of receiving states, $A \subseteq S$.



Activities of the machine:

Initially, the machine is in a defined initial state. At each step, reads a symbol from the input and enters a state which is given a value which, in the transition table corresponds to the current status and read back symbol. It continues reading the next symbol of the input, another transition by transition table etc. Depending on whether the machine stops after reading the input in a state that belongs to the set of recipient states, the automaton either the input accepted or not accepted. The set of all strings that the automaton accepts, form a regular language.

12. TURING MACHINES

Church (Church-Turing) thesis: Turing machines (and their equivalent systems), define their computing power what intuitively consider to be efficiently computable. For each algorithm there is an equivalent Turing machine.

Church's (Church-Turing) argument cannot be formally proved, but is supported by a number of arguments:

- Turing machines are very robust - various modifications do not alter their computing power
- It suggested a number of different computational models whose strength corresponds to Turing machines
- There is no known computational process which we would describe as effectively quantify and which would not be possible to implement the Turing machine

Turing machine is the theoretical model of a computer. It consists of several parts:

- Processor unit - finite automaton
- Program - Transition function rules
- Right-side endless tape - Used to record intermediate results

It is used for modeling algorithms in the theory of computability.

Turing's complete are the programming languages and computers that have the same computational power as Turing's machine

Definition:

$M = (Q, \Gamma, b, \Sigma, s, \delta, F)$	Turing machine
Q	Finite set of internal states
Γ	Finite alphabet of symbols on the tape
$b \in \Gamma$	An empty symbol is not part of the input string's alphabet
$\Sigma \subseteq \Gamma \setminus b$	Finite set of input symbols
$s \in Q$	Initial condition
$\delta: Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$	Transition function, L Move left to left, R Move the head to the right
$F \subseteq Q$	Set of final states
Configuration:	$\langle q, s, n \rangle \in Q \times \{yb^{\omega} \mid y \in \Gamma^{*}\} \times N_0$
q	Current status
s	The smallest continuous part of the tape containing the non-empty symbols

n Read head position (cell number)

If Q, Γ are disjoint sets, a compact shape can be used

Tape: 1234

Condition of the machine: q

Reader Head Position: 2

Configuration: 1q234

Initial configuration of TM input $w \in \Gamma^*$

$(s, wb^{\omega}, 0)$, Compact form: sw

Calculation method for Turing machine:

If the current status is the end state, it finishes the calculation

The read head reads one symbol from the cell on which it is currently located

If there is a transition defined in the transition function for the current state and a transition is defined for the symbol being read, then (in the case of multiple possible transitions in non-deterministic machines, one is chosen randomly):

It will change the status

The appropriate symbol is entered on the current head position

The head moves accordingly (does not shift)

There are various modifications of TM.

TM with the possibility to perform the calculation without head shifting

Transitional function extended by N – $\delta: Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R, N\}$

The N shift (no sweep of the read head) can be arranged on a conventional TS so that the read head reads the input symbol from the cell on which it is located, performs the appropriate operation and moves to the right (R). Now reads the symbol from the cell, writes the same symbol (i.e. does not change the symbol on which the head is located) and moves left (L). That's how we got the read head to the previous location and no other symbol was changed.

TM with two-sided endless tape

The tape is not left bounded

Possible move of the read head to the left of any configuration

N-tape TM

Reads from and writes to multiple tapes at once

The only change is in the transition function: $\delta: Q \times \Gamma^n \to Q \times (\Gamma \times \{L, R, N\})^n$

Non-deterministic TM (NTM)

Allows "multiple choice"

$$\delta: Q \times \Gamma \to 2^{Q \times \Gamma \times \{L,R,N\}}$$

Subroutines are used to facilitate the creation of complex TS. The subroutine is the set of states that contains the initial and final state. Usually it solves a partial problem in TS. In normal programming, it is the equivalent of a function.

The universal TS - U as input accepts the code of another TS T and the machine input word T. It decodes the transition function of the machine T and simulates the calculation of the machine. It can compute any partial recursive function (or is equivalent to universal partial recursive function), decides any recursive language and accepts any recursively recurring language.

13. COMPUTABILITY THEORY

Basic theory of computability question is: What is and what is not algorithmically computable (solve)? There are problems for which algorithms to solve them?

The problem is defined by:

- Name: XY
- Instance: what can be input?
- Result: output, input to output relationship

The problem is partial display type$\Sigma^* \rightarrow \Sigma^*$, valid for entry gives a valid output for invalid input special outcome "Incorrect input".

Problems have the following characteristics:

Algorithmic solvability

> For a given problem, we say that it is algorithmically solvable (or the respective (partial) view $\Sigma^* \rightarrow \Sigma^*$ is algorithmically computable) if there is an algorithm which is capable as an input to accept any instance of the problem and its calculation for any such entry ever ends, the outputs of the desired result.

Algorithmic decidability

> About the problem of Yes / No we say that the algorithm may decidable if there is an algorithm that is able to accept as input any instance of the problem and its calculation for any such entry ever ends, the output will be required answer Yes / No.

The problem is algorithmically solvable if there is an algorithm that for each particular task is always a problem in a specific time clearly determines the desired result. Decidability problems for a Yes / No

Partial decidability

> About the problem of Yes / No we will say that is partially decidable if there is an algorithm whose calculation is over just for those entries that match the instances of the problem with the answer Yes.
>
> The problem is partially decidable, if you know the algorithm stops and gives a result only if the answer to the question is yes. If the answer is No, so she'll never know, because the algorithm never stops.

Complexity theory provides us with interesting results. One of them is called Halting problem. We cannot construct an algorithm that would verify the finality of the general program of its run.

Complexity theory provides us with interesting hypothesis. One of them is the Church-Turing thesis. For each algorithm, there is an equivalent Turing machine.

Halting problem

Assignment: If you know the source code of the program and its input, decide whether the program stops, or whether to run forever without stopping.

In 1936 Alan Turing proved that there is no general algorithm that would solve the halting problem for all the inputs of all the programs there. Halting problem is therefore referred to as algorithmically undecidable problem.

Proof:

Undecidability of halting problem can be proved by contradiction.

Initial assumption: Assume that the halting problem can be decided. This means that there is a program stops (program, input), which is a universal solution to the problem of stopping - we assume, therefore, that if this program we will pass any program and its input, the program stops in a finite time returns the response such that if the calling program (input) after a finite number of steps over, then stops (program, input) returns YES, otherwise (if in Managing Calls program (input) trapped in a loop) returns Sun.

Subsequently Construct program Paradox (program) that calls Stops (software program), and if the call returned YES, deliberately zacyk divided, and if returned NE and ends immediately.

We ask, what is the result of a call Paradox (Paradox).

Assume for a moment that stops (Paradox, Paradox) returns YES. By definition, program Paradox then we know that Paradox (Paradox) loop forever. By definition, the program stops but, if the Paradox (Paradox) loop forever, then it must stop (Paradox, Paradox) returning Sun. We have come to dispute with.

On the other hand, suppose for a moment that stops (Paradox, Paradox) returns Sun. By definition, program Paradox then we know that Paradox (Paradox) stops. By definition, the program stops but, if the Paradox (Paradox) stops, then it must stop (Paradox, Paradox) return YES. Again, we came to the dispute.

Since we exhausted all possibilities and always reached the dispute shall be false initial premise. Consequently, the program stops (program, input), as defined, does not exist.

Church - Turing thesis

The hypothesis says that any possible calculation can be successfully carried out an algorithm running on the machine if there is sufficient time and memory.

The algorithm must meet the following requirements:

The algorithm consists of a finite number of instructions, which are precisely defined by using a finite number of symbols.

The algorithm always returns the result after a finite number of steps.

The algorithm can perform even a man with a pencil and paper.

Running the algorithm does not need human intelligence, except that which is necessary to understand and execute instructions.

Since every computer program can translate it into the language of a Turing machine, and vice versa, can be a thesis equivalent to formulate any commonly used programming language.

The programming language requires one of the following construct s (among others) that a Turing-complete (i.e. Equivalent Turing machine)

Cycle while -do.

Unlimited (at least in theory) recursion,

Conditional jump.

Common programming languages tend to have all three of these structures. Among the languages that are Turing- complete not include SQL (meaning no stored procedures.