Theoretical Foundations of Computer Science

Luku 0

Introduction

0.1 Course description

- 3 (+1?) cu
- possibly 1 cu "project work" about computational complexity and NP-complete problems
- Subject: computational problems vs. corresponding *formal languages* and mechanical models of solving problems vs. *automata* used for recognizing languages
- Background knowledge: basic mathematics
- Lectures: Mon, Tue 12-14 TD106
- Lecturer: Wilhelmiina Hämäläinen
  (whamalai@cs.joensuu.fi, meeting time Wed 14-15)

0.1.1 Exercise sessions

1. Wed 14-16 B179 Roman Bednari (English group)
2. Thu 14-16 B180 Wilhelmiina Hämäläinen (Finnish group)

0.1.2 How to perform?

3 choices:

1. Taking a part in problem-based learning and doing exercises, when the grade consists of exercises 25%, problem reports 50% and learning diary 25% of total points.

2. By doing exercises and performing two middle term exams, when exercises 25% and exams together 75% of total points.

3. By a separate exam, when the grade depends on only exam points.

0.1.3 Course material

- No one official course book, but a couple of recommended litterature

- "Lecture notes" cover the main topics of the whole course (the principal lecture notes in Finnish, but if possible, the main topics will be translated in English and appear in the course homepage http://www.cs.joensuu.fi/pages/whamalai/tepe/tfcp.html)

- It's recommendable to make your own notes! The lecture notes can be used as structure for your own material (also the text file available)

0.1.4 Litterature

Sipser, M. Introduction to the Theory of Computation.

Hopcroft, J.E., Motwani, R., Ulman, J.D. Introduction to Automata Theory, Languages, and Computation. Addison-Wesley, 2001. (or older edition)

Kinber, E., Smith, C. Theory of Computing. A Gentle Introduction. Prentice Hall, 2001. (Hyvin havainnollinen, kiltti johdatus aiheeseen :)

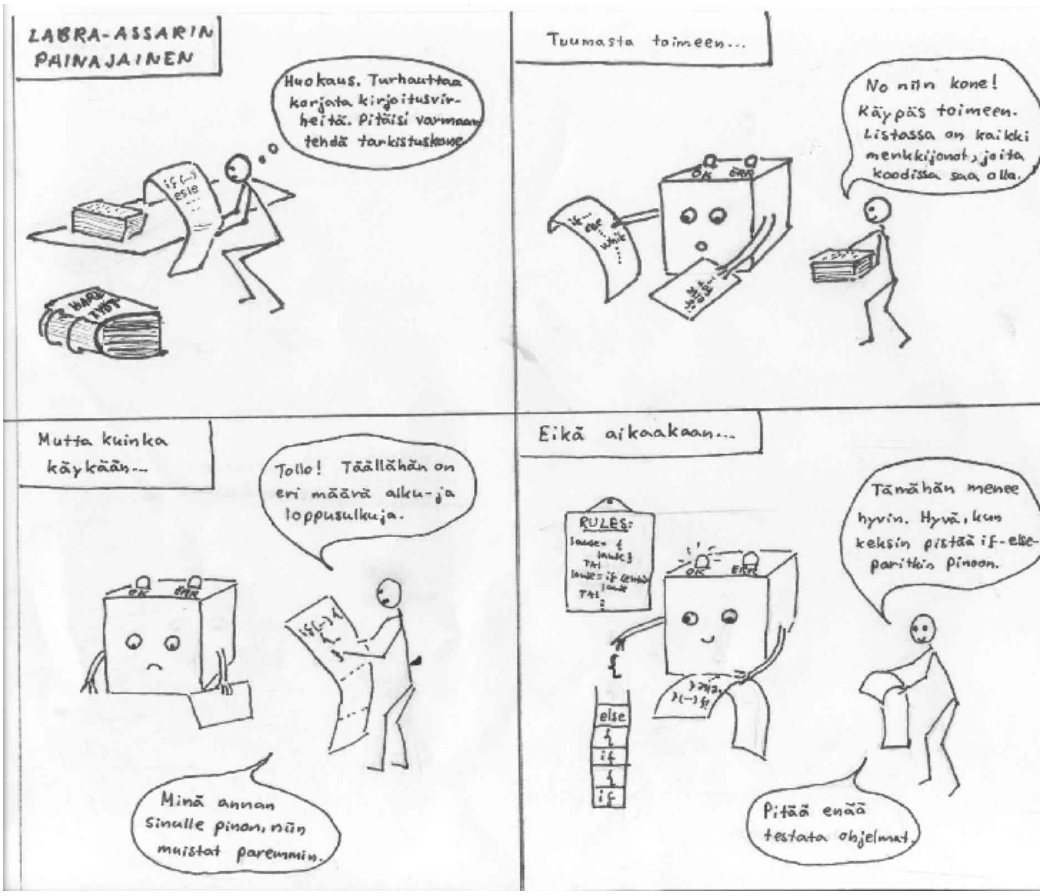
Lewis, H.R., Papadimitriou, C.H. Elements of the Theory of Computation, Second Edition, Prentice-Hall, 1998.

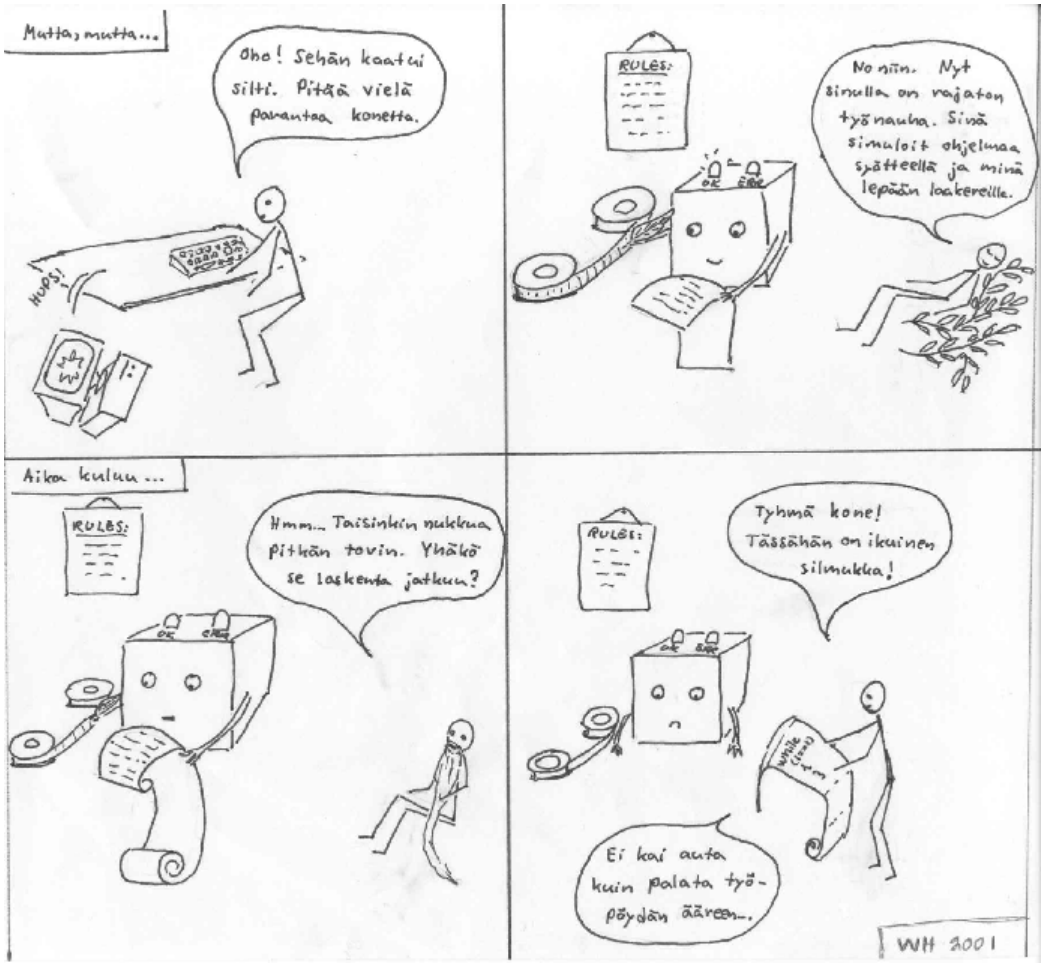
Wood, D. Theory of Computation, Harper & Row, 1987.

Sudkamp, T.A. Languages and Machines. An Introduction to the Theory of Computer Science.

0.1.5 Contents

Look at the following comic (The assistant's Nightmare):





1. "Frustrating to chech all misspellings. I should make a checking machine."

2. And so is done. "Well, Machine, let's work. All strings which may appear in the code are in the list."

3. But what happens...? "Grazy machine! Here is different number of beginning and end parenthesis. I'll give you a stack so you will remember better."

4. After a while. "This is going well. Good, that I invented to put also if-else structures into the stack. I'll just have to test the programs."

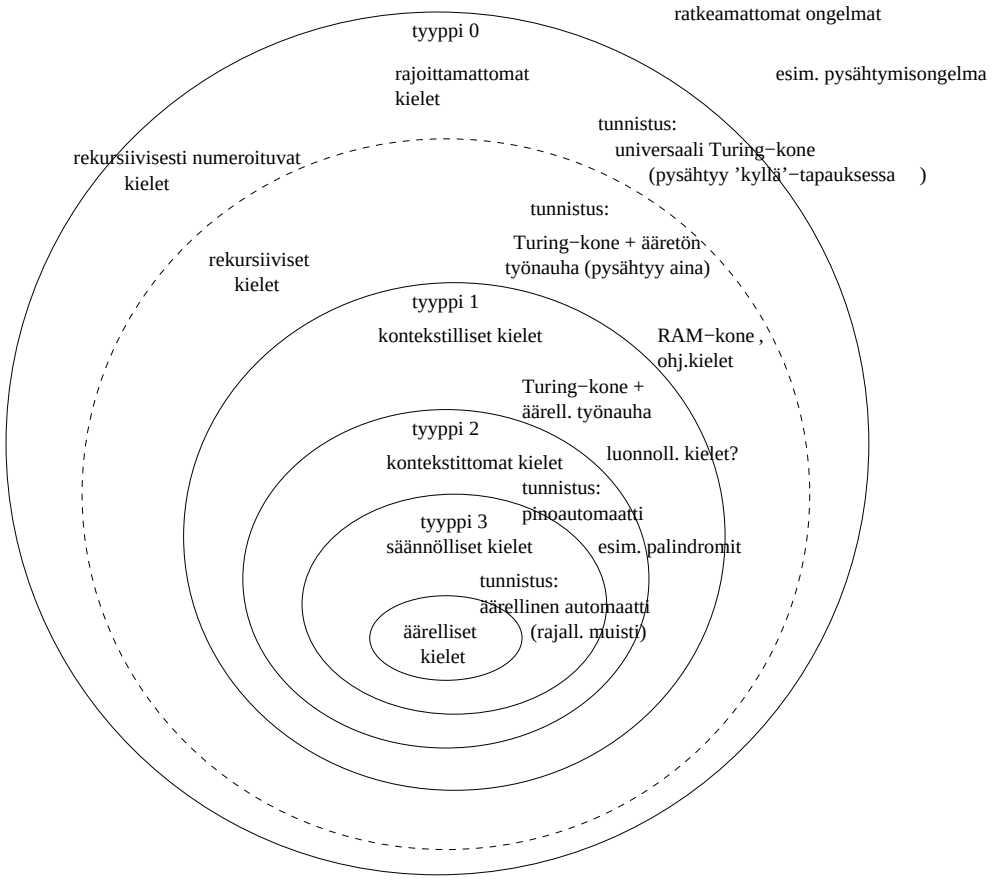
5. But: "Oh! It still fell down. I still must make the machine better."

6. "Ok, now you have an infinite tape. You'll simulate the program with the input while I lay on my leaves."

7. The time pasts. "Hmm, it seems I have slept quite a while. Is the computation still going on?"

8. "Stupid machine! There is an infinite loop. I guess I have to return back to work myself."

- v.s. Chomsky hierarchy of languages



Kuva 1: Chomsky language hierarchy + the class of recursive languages.

- type 3 *Regular languages* (special case finite languages)
- type 2 *Context-free languages*
- type 1 *Context-sensitive languages*
- type 0 *Unristricted languages* = *Recursive languages* + *Recursively countable languages*
- outside unsolvable problems

Preliminary contents:

1. Introduction
    - Introduction of the course
    - Mathematical concepts
    - Computational problems and solvability
2. Finite automata and regular languages
    - Regular expressions and languages
    - Deterministic finite automata
    - Minimizing automata
    - Undeterministic finite automata
    - Limitations of the regular languages
3. Grammars and parsing
    - Context-free grammars and languages
    - Recursive parsing
    - Attribute grammars
    - CYK-algorithm
    - Pushdown automata
4. Turing machines and unristricted languages
    - Turing machines
    - Extensions: multi track/tape Turing machines
    - undeterministic <-> deterministic machines
    - *Excursion: Context-sensitive languages and parsing the natural language
    - Self-respection of the machines, universal languages and machines
    - *Excursion: mechanical computability v.s. limits of human reasoning
    - Unsolvable problems
    - Comparing and reducing problems
5. Computational complexity
    - Time and space requirements
    - NP-completeness

0.2 Mathematical concepts

- logical symbols
- sets
- relations
- functions
- countability
- proof methods

0.2.1 Logical symbols

Let P and Q be *propositions* i.e. truth valued sentences, which describe some events. E.g. P="The Moon is cheese", Q="Napoleon lives in the Moon".

- $\neg P$: P is false (not P, $!P$)
- $P \vee Q$: either P or Q is true (or both) (in programming languages P or Q, $P||Q$)
- $P \wedge Q$: both P and Q are true (P and Q, $P\&\&Q$)
- $P \Rightarrow Q$: *implication* "if P, then Q" ($\equiv \neg P \vee Q$)
- $P \Leftrightarrow Q$: " *equivalence* "P if and only if Q" ($\equiv (P \Rightarrow Q) \wedge (Q \Rightarrow P)$)
- In addition we often need $\forall$ (*universal quantifier*, "for all") and $\exists$ (*existence quantifier*, "exists", "for some")
  E.g. $\forall x, x \in \mathbb{N}$ "for all natural numbers x ...", $\exists x, x \in \mathbb{N}$ "for some natural number x ..."

0.2.2 Sets

- *set*=a collection of *elements* or *members*
  e.g. $A = \{a_1, a_2, ..., a_n\}$ or $\Sigma = \{a, b, c, ..., z\}$
  mark $a_i \in A$ ("a_i belongs to set A")
- special case *empty set* $\emptyset$

- $A \subseteq B$: A is the *subset* of B

$$A \subseteq B \Leftrightarrow \forall x(x \in A \rightarrow x \in B)$$

proper subset $A \subset B$: $A \subseteq B \wedge A \neq B$

- $A \cup B$: *union* of A and B

$$A \cup B = \{x | x \in A \vee x \in B\}$$

- $A \cap B$: *intersection* of A and B

$$A \cap B = \{x | x \in A \wedge x \in B\}$$

- $A \setminus B$: A subtraction B

$$A \setminus B = \{x | x \in A \wedge x \notin B\}$$

- $\overline{A} = E \setminus A$: *complement* of A in E
- $A \times B$: *cartesian product* of A and B:

$$A \times B = \{(x, y) | x \in A \wedge y \in B\},$$

in which $< x, y >$ is *an ordered pair*.

- $\mathcal{P}(A)$: *the power set* of A

$$\mathcal{P}(A) = \{X | X \subseteq A\}$$

e.g. if $A = \{a, b, c\}$, then $\mathcal{P}(A) =$
$\{\emptyset, \{a\}, \{b\}, \{c\}, \{a, b\}, \{a, c\}, \{b, c\}, \{a, b, c\}\}$

0.2.3 Relations and functions

Relation

E.g. a (binary) relation between A and B is defined as a subset of $A \times B$:

$$R = \{(a, b) | a \in A \wedge b \in B \wedge R(a, b)\}$$

- E.g. A and B are natural numbers and R is a successor relation:

$$R(a,b), \text{ if and only if } b = a + 1.$$

  Then the relation consists of pairs $\{(0,1),(1,2),(2,3),...\}$.
- *inverse relation* of $R \subseteq A \times B$ $R^{-1} \subseteq B \times A$ is relation

$$R^{-1} = \{(b,a)|(a,b) \in R\}$$

Function

A relation between A and B $f \subseteq A \times B$ is a *function* or *mapping* from set A to set B, if the following conditions hold:

1. For each element of A there is a mapping in B i.e.

$$\forall x \in A \; \exists y \in B \; (y = f(x))$$

2. For each element of A there is only one mapping in B i.e.

$$\forall x_1, x_2 \in A(x_1 = x_2 \Rightarrow f(x_1) = f(x_2)$$

- Mark $y = f(x) \;\Leftrightarrow\; (x,y) \in f$.

Injection, surjection, bijection

Function $f : A \to B$ is

injection, if

$$\forall x_1, x_2 \in A(f(x_1) = f(x_2) \Rightarrow x_1 = x_2)$$

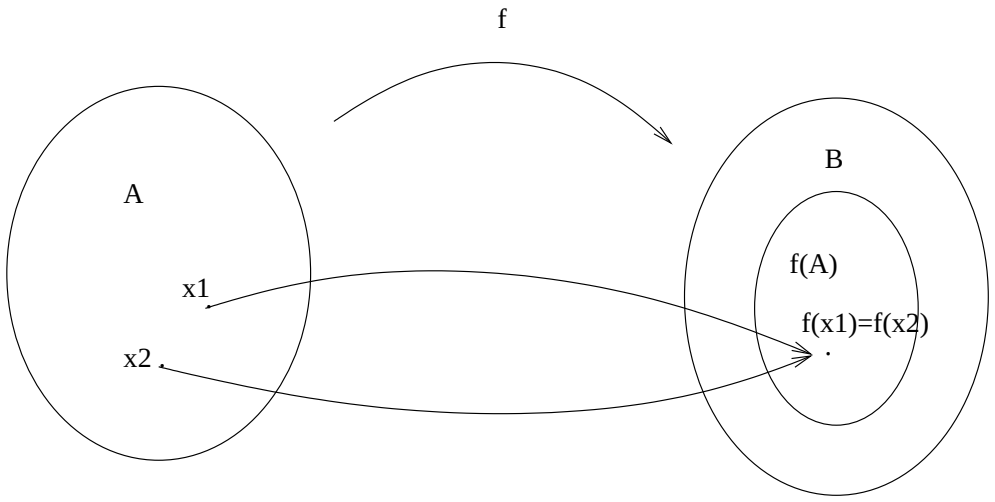
surjection, if

$$\forall y \in B \exists x \in A(y = f(x))$$

bijection, if f is both surjection and bijection

NOTICE! f is bijection $\Leftrightarrow$ f has inverse function

- e.g.1 $f : \mathbb{Z} \to \mathbb{Z}^+ \cup \{0\}$
  $f(x) = |x|$ surjection, not injection
- e.g.2 $f : \mathbb{Z}^+ \to \mathbb{Q}$
  $f(x) = \frac{1}{x}$ injection, not surjection
- e.g.3 $f : \mathbb{R}^+ \to \mathbb{R}^+$
  $f(x) = x^2$ bijection, inverse mapping $f(y) = \sqrt{y}$



Kuva 2: A=definition set, B=goal set, f(A)=value set

0.2.4 Countability

Set X is *countable*, if

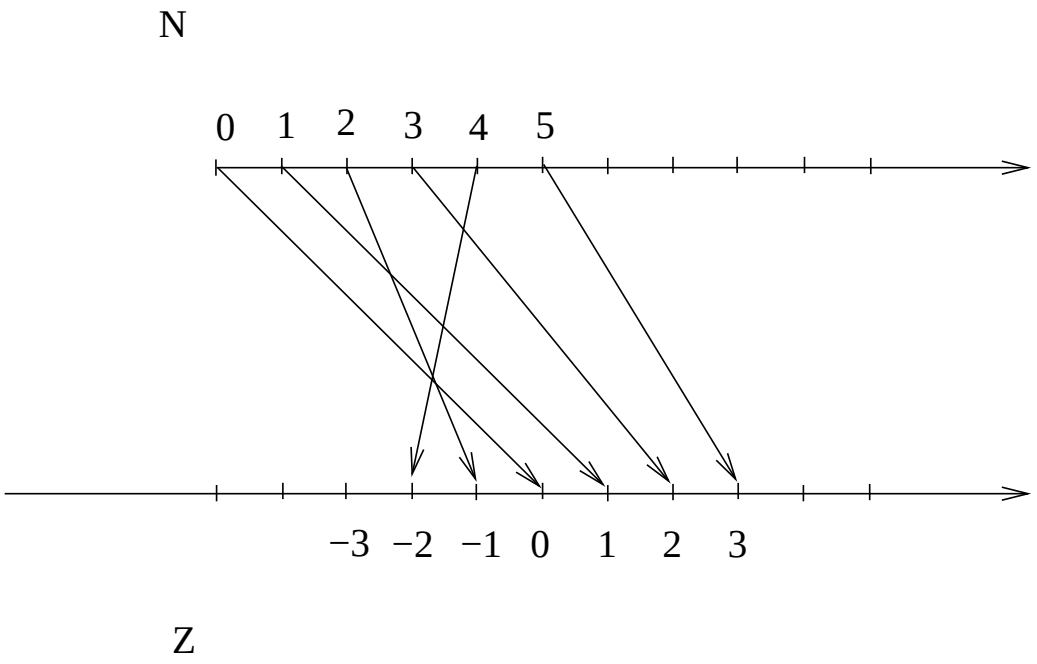
1. X is finite or
2. There exists a bijection $f : \mathbb{N} \to X$, for which

$$X = \{f(n) | n \in \mathbb{N}\}.$$

- intuitively: the elements of X can be ordered and indexed by natural numbers: $X = \{x_0, x_1, x_2, ...\}$.
- e.g. set of ood numbers $X = \{1, 3, 5, ...\}$ is countable, because we can define a bijection $f : \mathbb{N} \to X$

$$f(n) = 2n + 1$$

- If X is not countable it is *uncountable*.
- e.g. set of real numbers $\mathbb{R}$ and power set $\mathcal{P}(\mathbb{N})$ of natural numbers $\mathbb{N}$ are uncountable



Kuva 3: How to numerate integer numbers.

0.2.5 Proof methods

Mathematical induction

We want to show that $P(n)$ holds for all natural numbers.
Two parts:

1. Case $n = 0$: Prove that $P(0)$ holds.
2. Induction step: Prove that for all n $P(n) \rightarrow P(n+1)$

- e.g. Claim: $\Sigma_{i=1}^{n} i = \frac{n^2+n}{2}$ for all $n \geq 0$
  Proof:
  1. $n = 0$ $\Sigma_{i=1}^{n} i = 0 = \frac{0^2+0}{2}$
  2. Induction assumption: $\exists k \in \mathbb{N}$ such that the claim holds for all $n \leq k$
     Case $n = k+1$: $\Sigma_{i=1}^{k+1} i = 1 + 2 + 3 + ... + k + (k+1)$
     $\Sigma_{i=1}^{k} i = \frac{k^2+k}{2}$, so
     $\Sigma_{i=1}^{k+1} i = \frac{k^2+k}{2} + (k+1) = \frac{(k+1)^2+(k+1)}{2}$ $\square$

Undirect Proof (Contradiction method or Proof by antithesis)

We want to prove "if P, then Q". Let's suppose P and make an antithesis $\neg Q$. If we can conclude a contradiction, the claim is true.

- Notice! We don't know, what is the contradiction, we want to reach.
- Notice! Implication $P \Rightarrow Q$ is true in every case but when $P \wedge \neg Q$.
- e.g. Let's suppose U is an infinite set and S is a finite subset of it and T is the complement of S in U.

  Claim: T is infinite.

  Proof: Antithesis: T is finite. Because both S and T are finite, also U is finite, which means contradiction (U is infinite). $\square$
- How would you prove the claim: "In the world there are at least two people with exactly same number of hairs in their heads"?

Contraposition We want to prove "if P, then Q". Instead we prove an equivalent claim "if not Q, then not P" ($P \Rightarrow Q \equiv \neg Q \Rightarrow \neg P$)

- $\sim$ a special case of antithesis method: Suppose P and $\neg Q$ and try to conclude $\neg P$ – now we know, what is the contradiction we are looking for ($P \wedge \neg P$).

Proving existential and universal claims "There exists $x \in X$, for which..." and "For all $x \in X$ ..." can sometimes been proved directly.

- Existial claim $\exists x \in X P(x)$: construct such x (guess, produce, invent a producing algorithm etc.) Notice! You have to show that the wanted property really holds.
- e.g. "In the Himalaya there is a mountain, which is higher than any other mountain in the world."
- Universal claim $\forall x \in X P(x)$: select an arbitrary x from X and show that the wanted property $P(X)$ holds for it.
- e.g. Let $S = \{x \in R | (x^2 - 3x + 2 = 0)\}$ and $T = \{x \in R | 1 \leq x \leq 2\}$. Claim: $S = T$.

Counter examples

- Claim $\forall x \in X P(x)$ can be invalidated by giving any counter example in X
- e.g. claim "All cats are black."

Litterature

Solow, Daniel: How to read and write proofs. An introduction to mathematical thought process. John Wiley & Sons, 1982. (Easy reading guide book for constructing proofs!)

0.3 Excursion: Transfinite ordinal numbers

The transfinite ordinal numbers mean infinite numbers which still can be ordered. E.g. David Hilbert, Georg Cantor, Kurt Gödel and Rudy Rucker have studied problems dealing with such numbers.

ω Omega

ω or $\aleph_0$ (aleph-zero) is the greatest normal ordinal number, i.e. $\lim n$. ω can be defined: ω is the first such number a, for which $a + 1 = a$. E.g. $1 + \omega = \omega$, $2\omega = \omega$.

We can still define working sum and product operations for ω. Let's decide that $\omega + 1 =$ the follower of ω and $\lim_{n\to\omega}(\omega + n) = \omega + \omega = 2\omega$.

Now $\lim(\omega \times 2 + n) = \omega \times 3$, $\lim(\omega \times n) = \omega \times \omega = \omega^2$. In the same way $\omega^3, \omega^4, ..., \omega^\omega$.

ω^2 is the first such ordinal number a that $\omega + a = a$ and ω^ω is the first such ordinal number a that $\omega \times a = a$. ($\omega \times \omega^\omega = \omega^{\omega+1} = \omega^\omega$, because $1 + \omega = \omega$).

Notice! Sum and product operations for the transfinite numbers are not invertible!

ϵ_0 Epsilon-zero

Even greater numbers can be generated by nested exponents $\omega^{\omega^{\omega^{\omega^{...}}}}$. ϵ_0 is the first such ordinal number a that $\omega^a = a$.

We can describe ϵ_0 better by a new operation *tetration* ${}^a b$, which means a-times exponent b i.e. $b^{b^{...^b}}$, in which b appears a times. Very fast great numbers, i.e. ${}^3 10 = 10^{\text{milliard}}$, ${}^2\omega = \omega^\omega$, ${}^3\omega = \omega^{\omega^\omega}$. Now $\epsilon_0 = {}^\omega \omega$.

Even greater numbers by nested tetration operations!

$\aleph_1$ Aleph-one

The first ordinal number which is mahtavampi? than ω. I.e. there doesn't exist any bijection, which would function $\aleph_1$ elements into ω elements.

Hilbert's Hotel

Let's think about Hilbert's Hotel, in which there is infinite number of rooms, numbered by $0, 1, 2, 3, ...$ Hilbert's Hotel is such a strange hotel that even if it is full it can take new visitors. Let's suppose for example that there is one visitor in each room and a new visitor arrives to the hotel. How can we give her/him a room? Easily! We put the new visitor into room 0, which we get empty by moving the guest 0 into room 1, which we get empty by moving the guest 1 into room 2 and so on.

What about if there comes a group of infinite number of guests at once? Now we can put all the previous guests into even rooms and new guests into odd rooms.

In the same way we can fit ω^2, ω^ω or even ϵ_0 quests. However the capacity of the hotel has a limit: $\aleph_1$. $\aleph_1$ is the first such number that we cannot fit such number of guests into the hotel by any ordering.

Cantor's proof

Cantor showed 1873 that there are at least $\aleph_1$ points in the mathematical space. (Usually we say that the set of real numbers is uncountable). Cantor used a very smart technique in his proof, so called Cantor's diagonalization method:

Claim: The set of real numbers $\mathbb{R}$ is uncountable.

Proof: It's enough to study some subset of $\mathbb{R}$ and show that it is uncountable. Let's select interval $]0, 1[$ and make an antithesis:
Antithesis: The interval $]0, 1[$ is countable. It means that we can number all real numbers x, $0 < x < 1$ by natural numbers. Let's suppose that the numbering is done by a bijection r, which gives the real number x a number of order $r(x)$ ($r :]0, 1[\rightarrow \mathbb{N}$). Let's suppose that we can represent all real numbers of the interval $]0, 1[$ as an infinite matrix M, in which every natural number corresponds some real number represented with infinite precision. The beginning of the matrix could be following:

$r(1)$: .**1**41592...
$r(2)$: .3**3**3333...
$r(3)$: .71**8**281...
$r(4)$: .414**2**13...
$r(5)$: .5000**0**0...

Let's now construct a new real number in the following way: We read the digits in the diagonal of the matrix in order and change every digit to something else. I.e. if the desimal representation of the new number is $.d_1d_2d_3d_4...$ the ith desimal $d_i \neq M[i][i]$ (i.e. the ith desimal of the ith row).

The beginning of the number could be for example .02719.... However this number cannot appear in the matrix, because it differs from each number in the matrix: from number 1 in first digit, from number 2 in second digit, from number 3 in third digit and so on. So there cannot exist such ordering function r.

Litterature

Rucker, Rudy: White Light, or, What is Cantor's Continuum Problem? Ace Books, New York, 1982. (Imaginative science fiction novel, in which we play with infinities and also visit Hilbert's Hotel.)

Rucker, Rudy: Infinity and the Mind. The Science and Philosophy of Infinite. (An easy reading scientific book going on the themes of the White Light.)

Lem, Stanislaw: N. Ya. Vilenkin, Stories about Sets. Academic Press, New York, 1968. (A collection of shortstories, one of them about Hilbert's Hotel.)

Hofstadter, Douglas R.: Gödel, Esser, Bach: An Eternal Golden Braid. Vintage Books, New York, 1989. (Chapter XIII shows by Cantor's diagonalization argument that there exists problems which are algoritmically unsolvable. Also otherwise suitable reading for this course!)

0.4 Exercises about chapter 0

1. Pigeonhole Principle says: If you have more pigeons than pigeonholes, and each pigeon flies into some pigeonhole, then there must be at least one hole that has more than one pigeon.

   What happens, if you have as many pigeonholes as there are natural

numbers, and as many pigeons, as there are integers? What about, if you have as many pigeons as there are natural numbers, but each pigeon tries to make nest with every other pigeon into a different hole? (Only one nest can be made into one hole.)

2. How would you allocate $\omega \times \omega$ quests into ω rooms of Hilbert's Hotel?

3. In the quest book of Hilbert's Hotel there is only finite number of names in each page and new quests must always write their names into the next empty line. How many pages there must be in the book so that there is room for the new names (without reorganizing the names) as long as there is room in the hotel (maybe after reorganizing)?

4. Find the error in the following proof that $2 = 1$. Consider the equation $a = b$. Multiply both sides by a to obtain $a^2 = ab$. Subtract b^2 from both sides to get $a^2 - b^2 = ab - b^2$. Now factor each side, $(a-b)(a+b) = b(a-b)$, and divide each side by $(a-b)$, to get $a + b = b$. Finally, let a and b equal 1, which shows that $2 = 1$.

5. What is wrong in the following induction proof that all cats are of the same colour?

   Let n be the number of cats. If $n = 1$ the claim holds clearly (one cat is always of the same colour). Let's now suppose that for any group of n cats the claim holds. Then let's consider a group of $n + 1$ cats. By selecting any n cats from this group (which can be done in $n + 1$ different ways) we get by the induction assumption a group in which all the cats have the same colour. So all $n+1$ cats must be of the same colour.

6. Let X be a set and X the size of $n = |X|$. Prove by induction that the size of the powerset of X is $|\mathcal{P}(X)| = 2^n$.

7. Prove the following claim. If there are $n(n \geq 2)$ people in the party, then at least two people have equal number of friends in the party.

8. Prove by contraposition: If c is an odd integer number, then the equation $n^2 + n - c = 0$ doesn't have any integer solution for n.

Luku 1

Computational problems

- Computational problem $\sim$ any task which can be modelled such way that it can be solved by a computer
- E.g.: multiplying integers, ordering library cards, managing course database
- the solving program is one representation
- a more general representation is easier to analyze

1.0.1 Problem: The MIU-system

In the logic school of Kissastan the cat students study MIU-stystem, in which all the clauses are constructed from three letters: M, I and U. There is only one axiom, MI, in the system. New clauses (theorems) can be derived from the previous clauses (the axiom or theorems) by the following rules:

1. If you posses a string, whose last letter is I, you can add on a U at the end.
2. Suppose you have Mx (in which x can be any string, also an empty string). You can derive a string Mxx.
3. If III occurs in some string, you may replace it by U.
4. If UU occurs in some string, you can drop it from the string.

The rules can be applied freely, when ever they fit the axiom or already derived theorems, but you are not allowed to do anything else (that's why the system is called formal).

The cat students should derive from the axiom MI a theorem MU. Can you perform the reasoning?

So MIU-system can be thought as some kind of language game, in which we have an alphabet $\{M, I, U\}$ and a grammar to construct words. Let's represent the rules more formally:

$xI \rightarrow xU$
$Mx \rightarrow Mxx$
$xIIIy \rightarrow xUy$
$xUUy \rightarrow xy$,

in which x and y can be any strings.

Let's consider the set of all possible strings Σ^*, which consists of strings $\{\epsilon, M, I, U, MM, MI, MU, IM, II, IU, UM, UI, UU, MMM, MMI, MMU, MIM, ...\}$. Now we can make different questions about the strings. E.G. problem $\pi_1(x)$: "What strings can be produced from a given string x by the rules of the system?" or $\pi_2(x)$: "Can you produce string x from MI by the rules of the system?" The answer for the first question is a set of strings (or possibly an empty set), in fact a subset of Σ^*, while the answer of the latter one is simply "yes" or "no". Such yes/no-problems are called *decision problems*. Formally we can define a decision problem π as a mapping $\pi : \Sigma^* \rightarrow \{0, 1\}$. So it associates to each string of the alphabet either answer 1 or 0.

We can also ask, which strings of Σ^* does the decision problem π_A accept? (i.e. for which x $\pi(x) = 1$ or the inverted relation $\pi^{-1}(1) = x$?) The answer set A is called *a formal language* and the corresponding decision problem π_A the recognition problem of language A.

Problem: Which words do the following languages consist of?

- All strings which can be produced from M.
- All strings which can be produced from U.
- All strings which can be produced from MU

(The idea of the MIU-system is introduced in the book by Hofstadter, Gödel, Escher, Bach.)

1.0.2 Formalization

- the problem has potentially an infinite set of *cases* ("input")
- the solution is *an algorithm*, which associates to each case its *answer* ("output").
- e.g. multiplying integers
  - cases: all possible pairs of integers
  - an answer for a given pair: the product of the integers
  - solution of the problem: any algorithm for multiplying integers
- each case and its answer must be *finitely representable.*
- Computational problem = *a mapping from the set of cases to the set of answers*

1.0.3 Finite representation

- all information has to represented by bits in the last hand
- natural to allow also other symbols
- Definition: "finite representation" = *a string* in some alphabet.

1.0.4 Some concepts

- *Alphabet* a nonempty, finite set of *characters* or *symbols.* e.g. *binary alphabet* $\{0, 1\}$ and *latin alphabet* {A,B,...,Z}.
- *String* ordered queue of characters E.g. "01001", "000","LTE","XYZZY"
- *Length* of x: the number of characters. Mark $|x|$. E.g.

$$|01001| = |\text{XYZZY}| = 5, \qquad |000| = |\text{OTE}| = 3.$$

- *Empty string* ϵ, length $|\epsilon| = 0$.

- *Catenation* strings written together. E.g.

  (i) CAT^FISH = CATFISH;

  (ii) if $x = 00$ and $y = 11$, then $xy = 0011$ and $yx = 1100$;

  (iii) for all x $x\epsilon = \epsilon x = x$;

  (iv) for all x, y $|xy| = |x| + |y|$.

- The set of all strings of the alphabet Σ Σ^*. E.g. $\Sigma = \{0, 1\}$, $\Sigma^* = \{\epsilon, 0, 1, 00, 01, 10, \ldots\}$.

1.0.5 Decision problems and formal languages

- computational problem π is a mapping

$$\pi : \Sigma^* \to \Gamma^*,$$

  in which Σ and Γ are alphabets

- decision problems are a subclass of computational problems, for which the answer is "yes" or "no"

- the problem is of form $\pi : \Sigma^* \to \{0, 1\}$.

- e.g. "is a given number prime?" can be represented as a mapping in $\Sigma = \{0, 1, 2, \ldots, 9\}$

$$\pi : \Sigma^* \to \{0, 1\}, \qquad \pi(x) = \begin{cases} 1, & \text{if } x \text{ is prime;} \\ 0, & \text{if } x \text{ is not prime.} \end{cases}$$

- For each decision problem $\pi : \Sigma^* \to \{0, 1\}$ there is a set of strings

$$A_\pi = \{x \in \Sigma^* \mid \pi(x) = 1\},$$

  i.e. those cases for which the answer is "yes"

- for each set of string $A \subseteq \Sigma^*$ there is a decision problem

$$\pi_A : \Sigma^* \to \{0, 1\}, \qquad \pi_A(x) = \begin{cases} 1, & \text{if } x \in A; \\ 0, & \text{if } x \notin A. \end{cases}$$

- *formal language* in Σ = arbitrary set of strings $A \subseteq \Sigma^*$

- *recognition problem* of language A = decision problem π_A associated to A

- formal language decision problem

1.0.6 Solvability

- the program P *solves* the computational problem π, if for each input x the program P computes and outputs the value $\pi(x)$.
- Can all possible computational problems be solved by computer?
- No: the set of all possible strings (possible programs) is countable, but the set of all possible decision problems is uncountable (We cannot put $\aleph$ guests into ω rooms.)
- Notice! The result is *independent of the programming language used*!

Theorem 1 For any alphabet Σ the set of all strings Σ^* is countable.

Proof: Let $\Sigma = \{a_1, a_2, ..., a_n\}$. Let's fix an "alphabetic order" e.g. $a_1 < a_2 < ... < a_n$.

The strings of Σ^* can be ordered in (*canonical order*):

1. first list strings, whose length is 0 $(= \epsilon)$, then those, whose lengt is 1 $(= a_1, a_2, \ldots, a_n)$, then those, whose length is 2, and so on
2. in each length group the strings are listed in alphabetic order

- for each natural number n there is astring of Σ^* and vice versa $\rightarrow \Sigma^*$ is countable

Bijection $f : \mathbb{N} \rightarrow \Sigma^*$ is:

$$\begin{aligned}
0 &\mapsto \epsilon \\
1 &\mapsto a_1 \\
2 &\mapsto a_2 \\
&\vdots \quad\quad \vdots \\
n &\mapsto a_n \\
n+1 &\mapsto a_1 a_1 \\
n+2 &\mapsto a_1 a_2 \\
&\vdots \quad\quad \vdots \\
2n &\mapsto a_1 a_n \\
2n+1 &\mapsto a_2 a_1
\end{aligned}$$

$$\begin{array}{rcl} \vdots & & \vdots \\ 3n & \mapsto & a_2 a_n \\ \vdots & & \vdots \\ n^2+n & \mapsto & a_n a_n \\ n^2+n+1 & \mapsto & a_1 a_1 a_1 \\ n^2+n+2 & \mapsto & a_1 a_1 a_2 \\ \vdots & & \vdots \end{array} \qquad \square$$

Theorem 2 2 Any set of decision problems of Σ is uncountable

**Proof:* (Cantor's diagonalization argument.)
Lets mark the collection of all decision problems of Σ by Π

$$\Pi = \{\pi \mid \pi \text{ is mapping } \Sigma^* \to \{0,1\}\}.$$

Antithesis: Suppose that Π is countable, i.e. there exists a numbering

$$\Pi = \{\pi_0, \pi_1, \pi_2, \ldots\}.$$

Let the strings of Σ^*be in canonical order $x_0, x_1, x_2, \ldots$.
Let's construct a new decision problem $\hat{\pi}$:

$$\hat{\pi} : \Sigma^* \to \{0,1\}, \qquad \hat{\pi}(x) = \begin{cases} 1, & \text{jos } \pi_i(x_i) = 0; \\ 0, & \text{jos } \pi_i(x_i) = 1. \end{cases}$$

Because $\hat{\pi} \in \Pi$, $\hat{\pi} = \pi_k$ for some $k \in \mathbb{N}$.
Then

$$\hat{\pi}(x_k) = \begin{cases} 1, & \text{if } \pi_k(x_k) = \hat{\pi}(x_k) = 0; \\ 0, & \text{if } \pi_k(x_k) = \hat{\pi}(x_k) = 1. \end{cases}$$

CONTRADICTION. The claim that the set Π is countable, is false. $\square$

$\hat{\pi}$ $\searrow$	π_0	π_1	π_2	π_3	$\cdots$
x_0	$\not{0}^{1}$	0	0	1	$\cdots$
x_1	0	$\not{1}^{0}$	0	0	$\cdots$
x_2	1	1	$\not{1}^{0}$	1	$\cdots$
x_3	0	0	0	$\not{0}^{1}$	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

- we can solve only a small part of all existing computational problems by any programs
- all "strong enough" programming languages solve just the same class of solvable problems (Church–Turing thesis).
- most computational problems are *absolutely unsolvable.*
- also interesting practical problems
- e.g. *halting problem*: given a program P and its input x; we should decide, if the computation of P completes with input x, or does it stay in infinite loop

1.0.7 Excursion: The unsolvability of the halting problem

Interpreted in C-formalism: "There doesn't exist a total (always halting) C-program, which would solve, if the given C-program P halts with input w".

Let's suppose we could write a total C-funcion

int H(char *p, char *w),

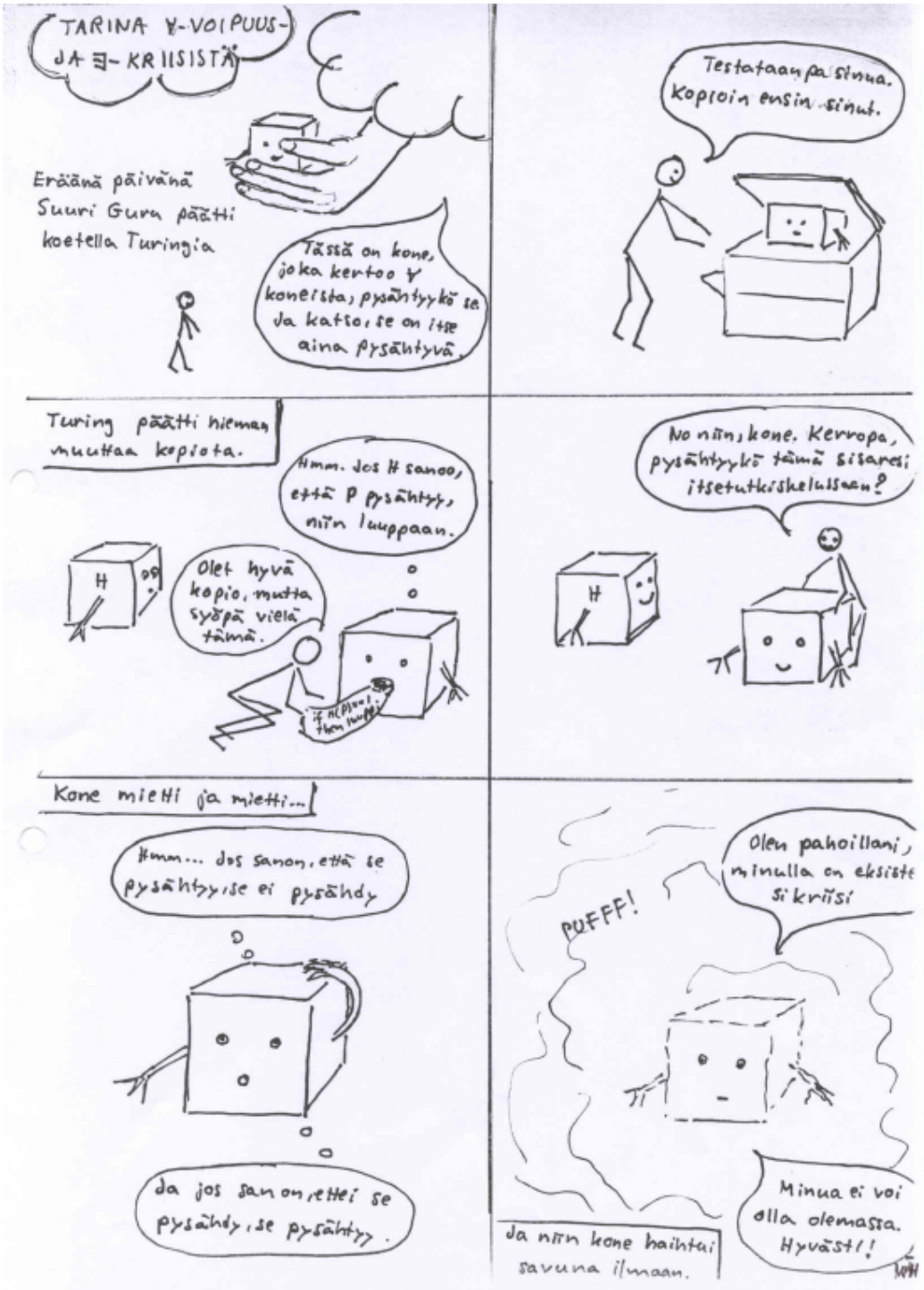
which gets value **1**, if the function represented by string p halts with input w, and **0** otherwise. Let's write another C-function $\hat{H}$:

void $\hat{H}$(char *p){
    if $H(p,p)$ while (1) ;
}

Let's mark the code of $\hat{H}$ by $\hat{h}$ and study the computation of $\hat{H}$ by its own description. Contradiction:

$$\hat{H}(\hat{h}) \text{ halts} \quad \Leftrightarrow \quad H(\hat{h},\hat{h}) = \mathbf{0} \quad \Leftrightarrow \quad \hat{H}(\hat{h}) \text{ doesn't halt.}$$

$\Rightarrow$ such total halting tester program H cannot exist.



The comic about Omnipotency and existency crisis.

One day the Great Guru decided to test Turing. "Here is a machine, which can tell about all machines, if they halt or not. And look! It will always halt itself."

"Let's study you.First I make a copy of you."

Turing decided to change the copy a little bit. "You are a very good copy, but still eat this." "Hmm. If H says that P halts I'll loop."

"Ok, machine. Tell me, if this sister of yours halts in her selfrespection."

The machine thought and thought. "Hmm... If I say that it halts it doesn't halt." "And if I say it doesn't halt it halts."

"I am sorry I have an existency crisis." "I cannot exist. Goodbye!" And so the machine vanished into air.

1.1 Exercises about chapter 1

1. Read the story about decision problems
   `http://www.cs.joensuu.fi/pages/whamalai/tfcs/story.html` and complete it!

2. Let's consider the logic school of Kissastan again (`http://www.cs.joensuu.fi/pages/whamalai/tfcs/problem0.html`).This time the topic is a more complicated MIAU-system, which consists of the following rules:

   xUAx $\rightarrow$ xAUy
   xUUx $\rightarrow$ xIUy
   x $\rightarrow$ MxM
   x $\rightarrow$ xUI
   xx $\rightarrow$ x
   xI $\rightarrow$ xUA

   The task is to show that even an empty string can create a proper miaow (MIAU) by the rules of the system!

3. Let's consider the alphabet $\Sigma = \{m, i, u\}$. The "powers" of the alphabet are defined in the following way:

   $\Sigma^0 = \{\epsilon\}$ (empty string)
   $\Sigma^{k+1} = \Sigma \times \Sigma^k = \{ax | a \in \Sigma \text{ and } x \in \Sigma^k\}$.

   E.g. $\Sigma^1 = \{m, i, u\}$, $\Sigma^2 = \{mm, mi, mu, im, ii, iu, um, ui, uu\}$. How many elements ("words") is there in Σ^n? What about in the whole language $\Sigma^* = \bigcup_{i=0}^{\infty} \Sigma^i$?