Aalto University
School of Science

CS-C2160 Theory of Computation

Lecture 1. Mathematical Preliminaries, Formal Languages

Pekka Orponen
Aalto University
Department of Computer Science

Spring 2022

Topics

- Mathematical preliminaries
  - Sets
  - Relations and functions
  - Equivalence relations
  - Proof by induction
- Automata and formal languages
  - Alphabets, strings and languages
  - Common notation
  - Automata and languages
  - Induction on strings

Material:

- Sections 1.1–1.6 in Finnish lecture notes (or Chapter 0 in the Sipser book)

Mathematical Preliminaries

1.1 Sets

- A *set* is a collection of elements.
- The elements can be given by enumerating, e.g.,

$$S = \{2, 3, 5, 7, 11, 13, 17, 19\}$$

or by some rule, e.g.,

$$S = \{p \mid p \text{ is a prime number and } 2 \leq p \leq 20\}.$$

- We write $a \in A$ to denote that element a belongs to set A. In the opposite case, we write $a \notin A$.

  For instance, $3 \in S$ and $8 \notin S$ for the set S above.
- An important special case is the *empty set* $\emptyset$ which does not have any elements.

- If all the elements of set A also belong to set B, we say that A is a *subset* of B (or that A is *included* in B) and write $A \subseteq B$.
  [Sometimes also notation $A \subset B$ is used in the literature.]
  If A is *not* a subset of B, we write $A \not\subseteq B$.
  For instance,
  $$\{2,3\} \subseteq S \qquad \text{and} \qquad \{1,2,3\} \not\subseteq S.$$
- Trivially, $\emptyset \subseteq A$ holds for all sets A.
- Two sets, A and B, are the same (denoted $A = B$), if they contain the same elements, i.e. if $A \subseteq B$ and $B \subseteq A$.
- If $A \subseteq B$ but $A \neq B$, we say that A is a *proper subset* of B and write $A \subsetneq B$.
  For instance, $\{2,3\} \subsetneq S$ and $\emptyset \subsetneq A$ whenever $A \neq \emptyset$.

- The elements in a set can also be other sets (in this case one often talks of a "family of sets"). For instance, we could define
  $$X = \{\emptyset, \{1\}, \{2\}, \{1,2\}\}.$$
- The family of sets formed by taking all the subsets of a base set A is called the *power set* over A and is denoted with $\mathcal{P}(A)$.
  As an example,
  $$\mathcal{P}(\{1,2\}) = \{\emptyset, \{1\}, \{2\}, \{1,2\}\}.$$
  [Because the power set over a base set with n elements contains 2^n elements, the notation 2^A is also used in the literature.]
- Observe that $A \subseteq B$ if and only if $A \in \mathcal{P}(B)$.
- Pay attention to the empty set:
  $$\emptyset \neq \{\emptyset\}, \quad \mathcal{P}(\emptyset) = \{\emptyset\}, \quad \mathcal{P}(\{\emptyset\}) = \{\emptyset, \{\emptyset\}\}.$$

The most common operations used to combine sets are:

Union
$$A \cup B = \{x \mid x \in A \text{ or } x \in B\},$$
E.g. $\{1,2,3\} \cup \{1,4\} = \{1,2,3,4\}$.

Intersection
$$A \cap B = \{x \mid x \in A \text{ and } x \in B\},$$
E.g. $\{1,2,3\} \cap \{1,4\} = \{1\}$.

Difference
$$A - B = \{x \mid x \in A \text{ and } x \notin B\},$$
E.g. $\{1,2,3\} - \{1,4\} = \{2,3\}$.

[The notation $A \setminus B$ is also used for set difference.]

The union and intersection operations are *associative*:
$$\begin{aligned} A \cup (B \cup C) &= (A \cup B) \cup C \\ A \cap (B \cap C) &= (A \cap B) \cap C \end{aligned}$$

and *commutative*:
$$\begin{aligned} A \cup B &= B \cup A \\ A \cap B &= B \cap A \end{aligned}$$

Furthermore, intersection *distributes* over union and vice versa:
$$\begin{aligned} A \cap (B \cup C) &= (A \cap B) \cup (A \cap C) \\ A \cup (B \cap C) &= (A \cup B) \cap (A \cup C) \end{aligned}$$

If all the sets we consider are subsets of some common “universal set” U, we call the difference $U - A$ the *complement* of A (in U) and denote it by $\bar{A}$.

The union, intersection and complement operations are related by the important *De Morgan laws*:

$$\begin{aligned} \overline{A \cup B} &= \bar{A} \cap \bar{B} \\ \overline{A \cap B} &= \bar{A} \cup \bar{B} \end{aligned}$$

In addition, the difference of two sets can be expressed using intersection and complementation:

$$A - B = A \cap \bar{B}$$

If the elements of a set family $\mathcal{A}$ are *indexed*, e.g.

$$\mathcal{A} = \{A_1, A_2, A_3, \dots\},$$

we define the following notation:

$$\bigcup_{i \geq 1} A_i = A_1 \cup A_2 \cup A_3 \cdots \quad \text{and} \quad \bigcap_{i \geq 1} A_i = A_1 \cap A_2 \cap A_3 \cdots$$

The indices do not need to be natural numbers. The *index set* can be any set I. In such a case, we use the notations

$$\mathcal{A} = \{A_i \mid i \in I\}$$

and

$$\bigcup_{i \in I} A_i, \qquad \bigcap_{i \in I} A_i.$$

1.2 Relations and functions

- Let A and B be sets.
- An *ordered pair* of two elements $a \in A$ and $b \in B$ is denoted by (a, b).
- Observe that for sets we have $\{a, b\} = \{b, a\}$, but $(a, b) \neq (b, a)$ whenever $a \neq b$.
- The *Cartesian product* of A and B is defined as

$$A \times B = \{(a, b) \mid a \in A \text{ and } b \in B\}.$$

Example

$\{1,2,3\} \times \{1,4\} = \{(1,1),(1,4),(2,1),(2,4),(3,1),(3,4)\}$
$\{a,b\} \times \mathbb{N} = \{(a,0),(b,0),(a,1),(b,1),(a,2),(b,2),\dots\}$
$\{a,b\} \times \emptyset = \emptyset$

- A *relation* R between a set A and a set B is a subset of the Cartesian product $A \times B$:

$$R \subseteq A \times B.$$

- When $(a, b) \in R$, we may also write $a \, R \, b$ and say that a is *in relation* R to b.

  This *infix* notation is used especially when the relation name is some mathematical symbol such as $\leq$, $\prec$, $\equiv$, $\sim$...
- If the *domain* A and *range* B sets of a relation R are the same, i.e. $R \subseteq A \times A$, we say that R is a relation *on* the set A.

- The *inverse* of a relation $R \subseteq A \times B$ is the relation $R^{-1} \subseteq B \times A$ defined as
$$R^{-1} = \{(b,a) \mid (a,b) \in R\}.$$
- If $R \subseteq A \times B$ and $S \subseteq B \times C$ are relations, then their *composite relation* $R \circ S \subseteq A \times C$ is defined as follows:
$$R \circ S = \{(a,c) \mid \exists b \in B \text{ such that } (a,b) \in R, (b,c) \in S\}.$$

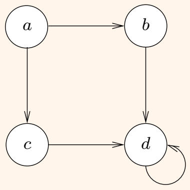
Especially when the sets A and B are finite, a relation $R \subseteq A \times B$ can be illustrated as a *directed graph*

- whose *vertices* are the elements in the sets A and B and
- there is an *arc* (directed edge) from vertex $a \in A$ to vertex $b \in B$ if and only if $(a,b) \in R$.

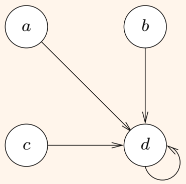
Example:

Consider the relation
$$R = \{(a,b),(a,c),(b,d),(c,d),(d,d)\}$$
on the set $A = \{a,b,c,d\}$.

R as a graph:



$R \circ R$ as a graph:



- A relation $f \subseteq A \times B$ is a *function* if each $a \in A$ is in relation f with exactly one $b \in B$. In this case, one uses the usual function notations $f : A \to B$ and $f(a) = b$.
- Everything above that applies to relations thus also applies to functions; however for historical reasons function composition is written from right to left.

  That is, if $f : A \to B$ and $g : B \to C$ are functions, then their function composition is defined as $(g \circ f)(a) = g(f(a))$, i.e. as the relation
$$g \circ f = \{(a,c) \mid \exists b \in B \text{ s.t. } f(a) = b, g(b) = c\}.$$

Let $f : A \to B$ be a function.

- f is *surjective*, or *onto*, if each $b \in B$ is an image of some $a \in A$.
- f is *injective*, or *one-to-one*, if each $a \in A$ is mapped to a distinct element in B, i.e., if $a \neq a' \Rightarrow f(a) \neq f(a')$.
- f is a *bijection* if it is both injective and surjective, i.e., if each $b \in B$ is the image of one, and only one, element $a \in A$.

1.3 Equivalence relations

- Equivalence relations are the mathematical formulation for the common idea that some objects are mutually equivalent / similar / indistinguishable with respect to some characteristic X.
- The equivalence relation based on a characteristic X partitions the set of objects into *equivalence classes* that correspond to the different values of characteristic X.
- Conversely, any partitioning Π of a set of objects into disjoint classes defines an abstract "characteristic" X_Π, where two objects are equivalent w.r.t. X_Π if and only if they belong to the same class in the partitioning Π.

The common idea of "equivalence" or "similarity" can be captured with three properties:

Definition 1.1

A relation $R \subseteq A \times A$ is

1. *reflexive* if $a \, R \, a$ holds for each $a \in A$;
2. *symmetric* if $a \, R \, b \Rightarrow b \, R \, a$ holds for all $a, b \in A$, and
3. *transitive* if $a \, R \, b \wedge b \, R \, c \Rightarrow a \, R \, c$ holds for all $a, b, c \in A$.

Definition 1.2 (Equivalence relation)

A relation $R \subseteq A \times A$ that is reflexive, symmetric and transitive is called an *equivalence relation*. The *equivalence class* (with respect to R) of an element $a \in A$ is

$$R[a] = \{x \in A \mid a \, R \, x\}.$$

Instead of R, equivalence relations are usually denoted by symbols such as $\sim$, $\equiv$, and $\simeq$.

Example:

Let

$$A = \{\text{all people born between 1921 and 2020}\}$$

and let $a \, R \, b$ hold if the persons a and b were born in the same year.

Then R is clearly an equivalence relation whose equivalence classes comprise all the persons who were born in a given year.

There are 100 equivalence classes, each corresponding to a year in $1921, \ldots, 2020$.

Proposition 1.3

Let $R \subseteq A \times A$ be an equivalence relation. Then for all $a, b \in A$ it holds that

$$R[a] = R[b] \quad \text{if and only if} \quad a \, R \, b.$$

Proposition 1.4

Let $R \subseteq A \times A$ be an equivalence relation. Then the equivalence classes of R partition A into mutually disjoint non-empty subsets, i.e.

- $R[a] \neq \emptyset$ for each $a \in A$,
- $A = \bigcup_{a \in A} R[a]$, and
- if $R[a] \neq R[b]$, then $R[a] \cap R[b] = \emptyset$, for all $a, b \in A$.

Correspondingly, each partitioning of a set A into disjoint non-empty subsets A_i, $i \in I$, defines an equivalence relation

$$a \sim b \quad \Leftrightarrow \quad a \text{ and } b \text{ belong to the same subset } A_i.$$

1.4 Proof by induction

The induction principle on natural numbers:

Let $P(k)$ be a predicate ($\sim$ property) with natural number argument k. If it holds that:

1. $P(0)$ and
2. for all $k \geq 0$:
$$P(k) \Rightarrow P(k+1),$$

then $P(n)$ holds for all $n \in \mathbb{N}$. □

Example

Claim. For all $n \in \mathbb{N}$ it holds that

$$P(n): \quad (1+2+\cdots+n)^2 = 1^3+2^3+\cdots+n^3.$$

Proof.

1. Induction basis $P(0): \quad 0^2 = 0.$

(continues)

2. Induction hypothesis and step: assume that, for some given $k \geq 0$, the formula

$$P(k): \quad (1+2+\cdots+k)^2 = 1^3+2^3+\cdots+k^3$$

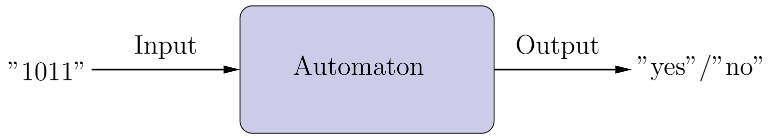
holds. Then also

$$\begin{aligned}
&(1+2+\cdots+k+(k+1))^2 \\
&= (1+\cdots+k)^2 + 2(1+\cdots k)(k+1) + (k+1)^2 \\
&= 1^3+\cdots+k^3 + 2\cdot\frac{k(k+1)}{2}\cdot(k+1) + (k+1)^2 \\
&= 1^3+\cdots+k^3 + k(k+1)^2 + (k+1)^2 \\
&= 1^3+\cdots+k^3 + (k+1)^3.
\end{aligned}$$

We have thus shown that whenever $P(k)$ holds, then also $P(k+1)$ holds, i.e., that $P(k) \Rightarrow P(k+1)$ for all $k \geq 0$. By induction, we can thus deduce that the formula $P(n)$ holds for all $n \in \mathbb{N}$. □

Automata and Formal Languages

Automata theory $\sim$ general theory of discrete I/O-mechanisms.



"1011" Input → Automaton → Output "yes"/"no"

Automata are *mathematical abstractions*. An automaton can be implemented in many ways, e.g., as a digital circuit, mechanical device or (most commonly) as a computer program.

In this course the focus is on automata whose

1. inputs are finite, discrete *strings*
2. outputs are of the form "accept"/"reject" ($\sim$ "yes"/"no" $\sim$ "input OK"/"input is invalid")

Many generalisations exist, e.g.

1. inputs can be infinite ($\rightarrow$ "reactive" systems, Büchi-automata)
2. outputs can be more complex ($\rightarrow$ Moore- and Mealy-machines, Turing machines computing functions)

1.5 Alphabets, strings and languages

An alphabet is a finite, non-empty set of *symbols* (also called *letters*). For instance,

- *the binary alphabet* $\{0,1\}$;
- *the Latin alphabet* $\{$A,B,...,Z$\}$.

A string is a finite sequence of symbols in some alphabet. As an example,

- "01001" and "0000" are strings over the binary alphabet,
- "TOC" and "XYZZY" are strings over the Latin alphabet, and
- the *empty string* contains no symbols and is denoted by ε.

The *length* of a string x is denoted by $|x|$ and is the number of symbols in it. E.g., $|01001| = 5$, $|\mathrm{TOC}| = 3$, and $|\varepsilon| = 0$.

- A basic operation between strings is *concatenation*, i.e., writing one string after another.
- For clarity, sometimes the symbol $^\frown$ is used for the binary concatenation operator.
- For instance,
  - SEA$^\frown$HORSE = SEAHORSE
  - If $x = 00$ and $y = 11$, then $xy = 0011$ and $yx = 1100$
  - For all strings x it holds that $x\varepsilon = \varepsilon x = x$
  - For all strings x, y, and z it holds that $(xy)z = x(yz)$
  - For all strings x and y, we have $|xy| = |x| + |y|$

- The set of all strings over an alphabet Σ is denoted by Σ^*.
- A subset $A \subseteq \Sigma^*$ is called a *(formal) language* over Σ.

Example:

If $\Sigma = \{0,1\}$, then $\Sigma^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, \ldots\}$.

Example:

If $\Sigma = \{0,1\}$, then the following are formal languages over Σ:

- $\emptyset$
- $\{\varepsilon, 0, 110110110011100\}$
- $\{0, 1, 11, 111, 1111, ...\}$
- $\Sigma^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000\ldots\}$

1.6 Automata and languages

- Let M be an automaton whose inputs are strings over an alphabet Σ, and the outputs are simply of form "input accepted" or "input rejected". (Abbreviated as $1/0$.)
- Denote the output of M on input x by $M(x)$, and the set of all inputs accepted by M by A_M, i.e.

$$A_M = \{x \in \Sigma^* \mid M(x) = 1\}.$$

  We say that M *recognises* the language $A_M \subseteq \Sigma^*$.
- A main idea in automata theory: *the structure of an automaton M is reflected in the properties of the language A_M*.
- Conversely: suppose that we are interested in an I/O-mapping $f : \Sigma^* \to \{0,1\}$. By studying the language

$$A_f = \{x \in \Sigma^* \mid f(x) = 1\}$$

  we may deduce *what kind of an automaton is needed to implement the function f*.

1.7 Common notation

In principle, we could use arbitrary symbols and notation for the concepts introduced above. However, for the sake of readability, it is customary to use the following fairly standard conventions:

Alphabets: Σ, Γ, ... (uppercase Greek letters). E.g. the binary alphabet $\Sigma = \{0,1\}$.

Size of an alphabet (or more generally any set): $|\Sigma|$.

Alphabet symbols: a, b, c, ... (lowercase Latin letters from the beginning of the alphabet). For instance, if $\Sigma = \{a_1, \ldots, a_n\}$ is an alphabet, then $|\Sigma| = n$.

Strings: u, v, w, x, y, ... (lowercase Latin letters from the end of the alphabet).

Concatenation of strings: $x^\frown y$ or simply xy.

Length of a string: $|x|$. For instance,

- $|abc| = 3$;
- If $x = a_1 \ldots a_m$ and $y = b_1 \ldots b_n$, then $|xy| = m + n$.

Empty string: ε.

String of n symbols a: a^n. As an example,

- $a^n = \underbrace{aa\ldots a}_{n\,\text{symbols}}$,
- $a^2b^3 = aabbb$, and
- $|a^ib^jc^k| = i + j + k$.

Repeating a string x k times: x^k. For instance,

- $(ab)^2 = abab$, and
- $|x^k| = k|x|$.

Set of all string over an alphabet Σ: Σ^*. As an example,

- $\{a,b\}^* = \{\varepsilon, a, b, aa, ab, ba, bb, aaa, aab, \ldots\}$.

1.8 Induction on strings

In automata theory it is common to construct things “by induction on the length of the string”. That is, one first defines an operation on the empty string ε (or on a single alphabet symbol). One then assumes that the operation is defined for all strings u of some length k and shows how to define it on strings of length $k+1$, i.e. strings of the form $w = ua$, where a is a single symbol.

Example:

Let Σ be an alphabet. The *reversal* w^R of a string $w \in \Sigma^*$ is defined inductively with the following rules:

1. $\varepsilon^R = \varepsilon$
2. if $w = ua$, $u \in \Sigma^*$, and $a \in \Sigma$, then $w^R = a\frown u^R$.

Such an inductive (“recursive”) definition can naturally be used in calculations, e.g.,

$$\begin{aligned}(011)^R &= 1\frown(01)^R &&= 1\frown(1\frown 0^R)\\ &= 11\frown(0\frown\varepsilon^R) &&= 110\frown\varepsilon^R\\ &= 110\frown\varepsilon &&= 110.\end{aligned}$$

However, induction is more often used to prove general properties of constructions, as shown on the next slide.

Claim. Let Σ be an alphabet. For all $x, y \in \Sigma^*$ it holds that $(xy)^R = y^R x^R$.

Proof. By induction on the length of y.

1. Base case when $|y| = 0$, i.e., $y = \varepsilon$: $(x\varepsilon)^R = x^R = \varepsilon^R x^R$.
2. Induction hypothesis and step: assume that the claim holds for all strings y of length k. Take any string $w = ya$ of length $k+1$, where $y \in \Sigma^*$, $|y| = k$ and $a \in \Sigma$. Then

$$\begin{aligned}(xw)^R &= (xya)^R &&\\ &= a\frown(xy)^R && \text{[definition of } R\text{]}\\ &= a\frown(y^R x^R) && \text{[induction hypothesis]}\\ &= (a\frown y^R)x^R && \text{[associativity of } \frown\text{]}\\ &= (ya)^R x^R && \text{[definition of } R\text{]}\\ &= w^R x^R. \quad \square &&\end{aligned}$$