

Finite automata and formal languages

*Notes based on lectures for DIT323 (Finite automata and formal languages)
at Gothenburg University, Spring 2024*

Sebastian Pålsson

Last updated: **January 24, 2024**

Contents

1 Introduction	2
1.1 Regular Expressions	2
1.2 Finite automata	2
1.3 Context-free grammars	3
1.4 Turing machines	3
1.5 Repetition of some classical logic	3
1.6 Repetition of some set theory	5
2 Proofs, induction & recursive functions	9

1 Introduction

A **finite automaton** is a computational model with a set of states, input symbols, transition rules, an initial state, and accepting states. It recognizes patterns and processes strings in a language by transitioning between states based on input.

Formal languages are abstract systems defined by rules to represent and analyze languages. They provide a precise framework for specifying syntax, semantics, and rules for generating valid strings within a language.

1.1 Regular Expressions

Regular expressions are concise patterns for searching and matching strings, widely used in text processing and pattern matching.

- Used in text editors
- Used to describe the lexical syntax of programming languages.
- Can only describe a limited class of “languages”.

Example

- A regular expression for strings of ones of even length: $(11)^*$
- A regular expression for some keywords: ***while*** | ***for*** | ***if*** | ***else***
- A regular expression for positive natural number literals (of a certain form): $[1-9][0-9]^*$

1.2 Finite automata

- Used to implement regular expression matching.
- Used to specify or model systems.
 - One kind of finite automaton is used in the specification of TCP.
- Equivalent to regular expressions.

Example

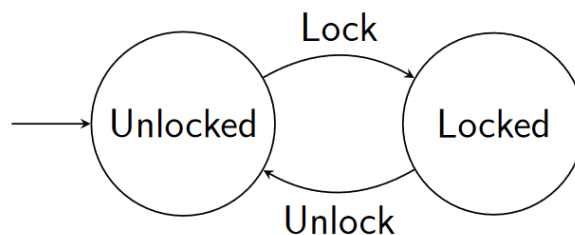


Figure 1: Model of a lock

- The states are a kind of memory.
- Finite number of states \Rightarrow finite memory.

1.3 Context-free grammars

Used to describe the syntax of programming languages.

- More general than regular expressions.
- Used by parser generators. (Often restricted.)

```

1 Expr ::= Number
2       | Expr Op Expr
3       | '(' Expr ')'
4 Op ::= '+' | '-' | '*' | '/'

```

1.4 Turing machines

A Turing machine is an abstract model of computation with a tape, read/write head, and rules. It serves as a foundational concept in the study of algorithms and computability.

- Unbounded memory: an infinite tape of cells.
- A read/write head that can move along the tape.
- A kind of finite state machine with rules for what the head should do.

It is equivalent to a number of other models of computation.

1.5 Repetition of some classical logic

1.5.1 Propositions

A **proposition** is a statement that is either true or false.

Example

- The sky is blue.
- The sky is green.
- $1 + 1 = 2$.
- $1 + 1 = 3$.

It may not always be known what the truth value (\top or \perp) of a proposition is.

1.5.2 Connectives

Logical connectives are operators that combine propositions to form new propositions.

$p \wedge q$	conjunction
$p \vee q$	disjunction
$\neg p$	negation
$p \Rightarrow q$	implication
$p \Leftrightarrow q$	equivalence

Truth tables for these connectives:

p	q	$p \wedge q$	$p \vee q$	$\neg p$	$p \Rightarrow q$	$p \Leftrightarrow q$
\top	\top	\top	\top	\perp	\top	\top
\top	\perp	\perp	\top	\perp	\perp	\perp
\perp	\top	\perp	\top	\top	\top	\perp
\perp	\perp	\perp	\perp	\top	\top	\top

Note that $p \Rightarrow q$ is true if p is false.

1.5.3 Validity

A proposition is *valid*, or a *tautology*, if it is satisfied for all assignments of truth values to its variables.

Example

- $p \Rightarrow p$ is valid.
- $p \vee \neg p$ is valid

1.5.4 Equivalence

Two propositions are *equivalent* if they have the same truth value for all assignments of truth values to their variables. (they have the same truth table)

Example

- $p \Rightarrow q$ and $\neg p \vee q$ are equivalent.
- $p \wedge q$ and $q \wedge p$ are equivalent.

1.5.5 Predicates

A predicate is, roughly speaking, a function to propositions.

Example

- $P(n) = "n \text{ is a prime number}"$
- $Q(a, b) = "(a + b)^2 = a^2 + 2ab + b^2"$

1.5.6 Quantifiers

Universal quantification and **existential quantification** are used to express statements about all or some elements in a set.

Example

- $\forall n \in \mathbf{N} : P(n)$ means that $P(n)$ is true for all natural numbers n .
- $\exists n \in \mathbf{N} : P(n)$ means that $P(n)$ is true for some natural number n .

1.6 Repetition of some set theory

A set is roughly speaking a collection of elements.

1.6.1 Defining sets

- $A = \{1, 2, 3\}$ means that A is the set containing the elements 1, 2, and 3.
- $B = \{x \in \mathbb{N} \mid x > 0\}$ means that B is the set of all natural numbers x such that $x > 0$.
- $C = \{x \in \mathbb{N} \mid \exists y \in \mathbb{N} : x = 2y\}$ means that C is the set of all natural numbers x such that there exists a natural number y such that $x = 2y$. (the set of all even natural numbers)

1.6.2 Members, subsets, and equality

- $x \in A$ means that x is an element of A .
- $A \subseteq B$ means that A is a subset of B .
- $A = B$ means that A and B are equal.

1.6.3 The empty set

- \emptyset is the empty set.
- $\forall x : \neg x \in \emptyset$.

1.6.4 Set operations

Union, intersection and set difference

- $A \cup B$ is the union of A and B . (the set of all elements that are in A or B)
- $A \cap B$ is the intersection of A and B . (the set of all elements that are in A and B)
- $A \setminus B = A - B$ is the set difference of A and B . (the set of all elements that are in A but not in B)

Complement

- \overline{A} is the complement of A . (the set of all elements that are not in A)

Cartesian product

- $A \times B$ is the Cartesian product of A and B . (the set of all pairs (a, b) where $a \in A$ and $b \in B$)

Example

- $\mathbb{N} \times \mathbb{N}$ is the set of all pairs of natural numbers.
- $\mathbb{N} \times \mathbb{N} \times \mathbb{N}$ is the set of all triples of natural numbers.

Power set

- $\wp(A) = \{A \mid A \subseteq S\}$ is the power set of A . (the set of all subsets of A)

Example

- $\wp(\{1, 2\}) = \{\emptyset, \{1\}, \{2\}, \{1, 2\}\}$

Set of all finite subsets

- $\text{Fin}(A) = \{A \mid A \subseteq S, A \text{ is finite}\}$ is the set of all finite subsets of A .

Example

- $\text{Fin}(\{1, 2\}) = \{\emptyset, \{1\}, \{2\}, \{1, 2\}\}$
- $\text{Fin}(\mathbb{N}) = \wp(\mathbb{N})$

1.6.5 Relations

Relations define connections between elements of sets. A binary relation is a subset of the Cartesian product of two sets, often denoted as $R \subseteq A \times B$. Common types include reflexive, symmetric, and transitive relations, capturing different aspects of element connections.

- A binary relation R on A is a subset of $A^2 = A \times A : R \subseteq A^2$
- Notation: xRy same as $(x, y) \in R$
- Can be generalised from $A \times A$ to $A \times B \times C \times \dots$

Some binary relational properties

For $R \subseteq A \times B$:

- Total (left-total): $\forall x \in A : \exists y \in B : xRy$
- Functional/deterministic: $\forall x \in A : \forall y, z \in B : xRy \wedge xRz \Rightarrow y = z$

For $R \subseteq A^2$:

- Reflexive: $\forall x \in A : xRx$
- Symmetric: $\forall x, y \in A : xRy \Rightarrow yRx$
- Transitive: $\forall x, y, z \in A : xRy \wedge yRz \Rightarrow xRz$
- Antisymmetric: $\forall x, y \in A : xRy \wedge yRx \Rightarrow x = y$

Partial orders

A *partial order* is a relation that is reflexive, antisymmetric, and transitive.

Equivalence relations

An *equivalence relation* is a relation that is reflexive, symmetric, and transitive.

1.6.6 Functions

Relation between two sets, denoted as $f : A \rightarrow B$, where A is the *domain* (set of inputs) and B is the *codomain* (set of possible outputs). Every element in the *domain* is associated with a unique element in the *codomain*. If (x, y) is in the function, it means that the input x is associated with the output y .

- Sometimes defined as the set of total and functional relations $f \subseteq A \times B$
- Notation $f(x) = y$ same as $(x, y) \in f$
- If the requirement of totality is dropped, we get the set of partial functions, $A \rightharpoonup B$
- The *image* is the set of all outputs of the function, $\{y \in B \mid x \in A, f(x) = y\}$

Identity, composition

- The *identity function* $\text{id}_A : A \rightarrow A$ is defined as $\text{id}_{A(x)} = x$
- For functions $f \in B \rightarrow C$ and $g \in A \rightarrow B$ the *composition* of $f \circ g \in A \rightarrow C$ is defined by $(f \circ g)(x) = f(g(x))$

Injectivity

An *injection* is a function $f : A \rightarrow B$ such that $\forall x, y \in A : f(x) = f(y) \Rightarrow x = y$

- Every input is mapped to a unique output.
- A is at most as large as B.
- Holds if f has a left inverse $g \in B \rightarrow A : g \circ f = id$

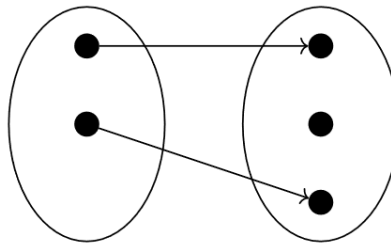


Figure 2: Injective function

Surjectivity

A *surjection* is a function $f : A \rightarrow B$ such that $\forall y \in B : \exists x \in A : f(x) = y$

- The function “targets” every element in the *codomain*
- A is at least as large as B.
- Holds if f has a right inverse $g \in B \rightarrow A : f \circ g = id$

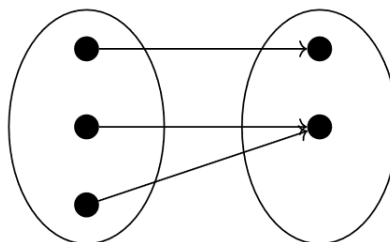


Figure 3: Surjective function

Bijectivity

A *bijection* is a function $f : A \rightarrow B$ such that $\forall y \in B : \exists! x \in A : f(x) = y$. In simple terms, it is both injective and surjective.

- A and B are of the same size.
- Holds iff f has left and right inverse $g \in B \rightarrow A$

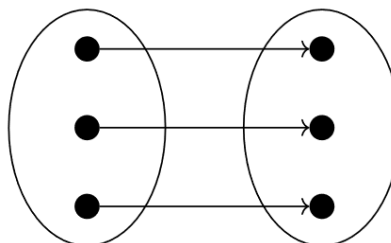


Figure 4: Bijective function

1.6.7 Partitions

A *partition* $P \subseteq \wp(A)$ of a set A is a set of non-empty subsets of A such that every element in A is in exactly one of the subsets.

- Every element is non-empty: $\forall X \in P : X \neq \emptyset$
- The elements cover A : $\bigcup_{B \in P} B = A$
- The elements are mutually disjoint: $\forall B, C \in P : B \neq C \Rightarrow B \cap C = \emptyset$

Example

- $P = \{\{1, 2\}, \{3, 4\}\}$ is a partition of $A = \{1, 2, 3, 4\}$
- $P = \{\{1, 2\}, \{3, 4\}, \{5\}\}$ is not a partition of $A = \{1, 2, 3, 4\}$

1.6.8 Equivalence classes

Given a set A and an equivalence relation $R \subseteq A \times A$, the *equivalence class* of an element $a \in A$ is the set of all elements in A that are equivalent to a .

Definition 1.6.8.1

The equivalence classes of an equivalence relation R on A :

$$[x]_R = \{y \in A \mid xRy\} \quad [1]$$

1.6.9 Quotients

Given a set A and an equivalence relation $R \subseteq A \times A$, the quotient of A by R is the set of all equivalence classes of R .

Definition 1.6.9.1

The quotient of A by R is the set of all equivalence classes of R :

$$\frac{A}{R} = \{[x]_R \mid x \in A\} \quad [2]$$

Example

Can one define $\mathbb{Z} = \mathbb{N}^2$ with the intention that (m, n) stands for $m - n$?

No, $(0, 1)$ and $(1, 2)$ would both represent -1 .

Instead we can use the quotient set:

$$\mathbb{Z} = \mathbb{N}^2 \setminus \sim \quad [3]$$

where

$$(m_1, n_2) \sim_{\mathbb{Z}} (m_2, n_2) \Leftrightarrow m_1 + n_1 = m_2 + n_2 \quad [4]$$

2 Proofs, induction & recursive functions