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 25, 2024**

Contents

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

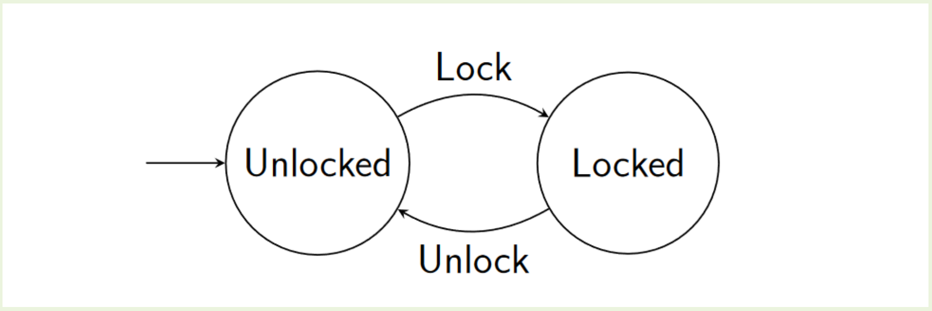

Figure 1: Model of a lock

- The states are a kind of memory.
- Finite number of states $\Longrightarrow$ 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.)

```
Expr ::== Number
    | Expr Op Expr
    | '(' Expr ')'
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 $\bot$) of a proposition is.

1.5.2 Connectives

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

$\mathbf{p} \land \mathbf{q}$	conjuction
$\mathbf{p} \lor \mathbf{q}$	disjunction
$\neg\mathbf{p}$	negation
$\mathbf{p} \Rightarrow \mathbf{q}$	implication
$\mathbf{p} \Leftrightarrow \mathbf{q}$	equivalence

Truth tables for these connectives:

p	q	p ∧ q	p ∨ q	¬p	p ⇒ q	p ⇔ q
⊤	⊤	⊤	⊤	⊥	⊤	⊤
⊤	⊥	⊥	⊤	⊥	⊥	⊥
⊥	⊤	⊥	⊤	⊤	⊤	⊥
⊥	⊥	⊥	⊥	⊤	⊤	⊤

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 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

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

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

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

1.6.3 The empty set

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

1.6.4 Set operations

Union, intersection and set difference

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

Complement

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

Cartesian product

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

Example

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

Power set

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

Example

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

Set of all finite subsets

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

Example

- $\mathbf{Fin}(\{\boldsymbol{1, 2}\}) = \{\emptyset, \{\boldsymbol{1}\}, \{\boldsymbol{2}\}, \{\boldsymbol{1, 2}\}\}$
- $\mathbf{Fin}(\boldsymbol{N}) = \wp(\boldsymbol{N})$

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 $A \times B \times C \times \cdots$

Some binary relational properties

For $R \subseteq A \times B$:

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

For $R \subseteq A^2$:

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

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 $f : A \to B$, 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* $\mathrm{id}_A : A \to A$ is defined as $\mathrm{id}_{A(x)} = x$
- For functions $f \in B \to C$ and $g \in A \to B$ the *composition* of $f \circ g \in A \to C$ is defined by $(f \circ g)(x) = f(g(x))$

Injectivity

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

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

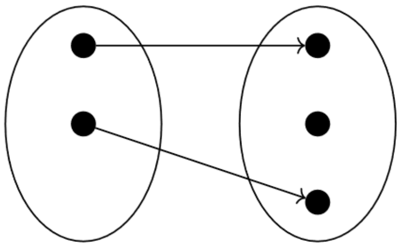

Figure 2: Injective function

Surjectivity

A *surjection* is a function $f : A \to B$ 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 $g \in B \to A : f \circ g = id$

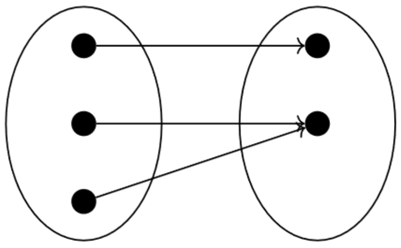

Figure 3: Surjective function

Bijectivity

A *bijection* is a function $f : A \to B$ 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 $g \in B \to A$

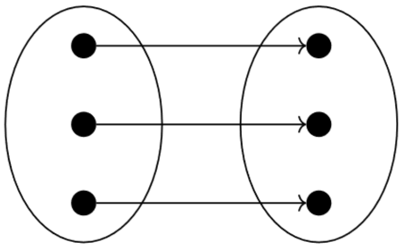

Figure 4: Bijective function

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:

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

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:

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

where

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

2 Proofs, induction & rectursive functions

3 Nondeterministic finite automata

3.1 NFAS

LiKE DFAs but with multiple transisitons may be possible.

- Can be in multiple states at once
- Can be easier to “program”.
- Can be much more compact.

An NFA can be given by a 5-tuple (Q)