# Chapter 1

# Basic Definitions

The first chapter defines fundamentals of this thesis and notation used later on.

## 1.1 General Mathematical Terms

As our main focus is $\omega$-words, we will require a small extension of natural numbers into the transfinite realm.

### 1.1.1 Sets and Functions

**Definition 1.1.1.** The *natural numbers* $\mathbb{N} = \{0, 1, 2, \dots\}$ are the set of all non-negative integers. We define $0 := \emptyset$, $1 := \{0\}$, $2 := \{0, 1\}$, and so forth.

The value $\omega$ denotes the "smallest" infinity, $\omega := \mathbb{N}$. For all natural numbers, we write $n < \omega$ and $\omega \not< \omega$. Also, we sometimes use the convention $n + \omega = \omega$.

We denote the set $\mathbb{N} \cup \{\omega\}$ by $\mathbb{N}_\omega$.

**Definition 1.1.2.** Let $X$ and $Y$ be two sets. We use the usual definition of union ($\cup$), intersection ($\cap$), and set difference ($\setminus$). If some domain ($X \subseteq D$) is clear in the context, we write $X^\complement = D \setminus X$.

We use the cartesian product $X \times Y = \{(x, y) \mid x \in X, y \in Y\}$.

We write $X^Y$ for the set of all functions with domain $Y$ and range $X$. If we have a function $f : D \to \{0, 1\}$, then we sometimes implicitly use it as a set $X \subseteq D$ with $x \in X$ iff $f(x) = 1$. In particular, $2^Y$ is the powerset of $Y$.

**Definition 1.1.3.** Let $f : D \to R$ be a function and let $X \subseteq D$ and $Y \subseteq R$. We describe by $f(X) = \{f(x) \in R \mid x \in X\}$ and $f^{-1}(Y) = \{x \in D \mid \exists y \in Y : f(x) = y\}$.

**Definition 1.1.4.** Let $X \subseteq D$ be a set. For $D' \subseteq D$, we define $X \restriction_{D'} = X \cap D$. In particular, we use this notation for relations, e.g. $R \subseteq \mathbb{N} \times \mathbb{N}$ and $R \restriction_{\{0\} \times \mathbb{N}}$.

For a function $f : D \to R$, we write $f \restriction_{D'}$ for the function $f' : D' \to R, x \mapsto f(x)$.

### 1.1.2 Relations and Orders

**Definition 1.1.5.** Let $X$ be a set. We call a set $R \subseteq X \times X$ a *relation* over $X$. $R$ is

- *reflexive*, if for all $x \in X$, $(x, x) \in R$.
- *irreflexive*, if for all $x \in X$, $(x, x) \notin R$.
- *symmetric*, if for all $(x, y) \in R$, also $(y, x) \in R$.
- *asymmetric*, if for all $(x, y) \in R$, $(y, x) \notin R$.
- *transitive*, if for all $(x, y), (y, z) \in R$, also $(x, z) \in R$.
- *total*, if for all $x, y \in X$, $(x, y) \in R$ or $(y, x) \in R$ is true.

We call $R$

- a *partial order*, if it is irreflexive, asymmetric, and transitive.
- a *total order*, if it is a partial order and total.
- a *preorder*, if it is reflexive and transitive.
- a *total preorder*, if it is a preorder and total.
- an *equivalence relation*, if it is a preorder and symmetric.

If $R$ is a partial order or a preorder, we call an element $x \in X$ *minimal* (w.r.t. $R$), if for all $y \in X$, $(y, x) \in R$ implies $(x, y) \in R$. Similarly, we call it *maximal*, if for all $y \in X$, $(x, y) \in R$ implies $(y, x) \in R$.

We call $x$ the *minimum* of $R$ if for all $y \neq x$, $(y, x) \in R$. We write $x = \min_R X$.

**Definition 1.1.6.** Let $R$ be a partial order over $X$. We call a set $S \subseteq Y$ an *extension of $R$ to $Y$* if $X \subseteq Y$, $R \subseteq S$, and $S$ is a partial order over $Y$. We use the same notation for total orders, preorders, and total preorders.

**Definition 1.1.7.** Let $R$ be an equivalence relation over $X$. $R$ implicitly forms a partition of $X$ into *equivalence classes*. For an element $x \in X$, we call $[x]_R := \{y \in X \mid (x, y) \in R\}$ the equivalence class of $x$.

## 1.2 Words and Languages

**Definition 1.2.1.** A non-empty set of symbols can be called an *alphabet*, which we will denote by a variable $\Sigma$ most of the time. As symbols, we usually use lower case letters, i.e. $a$ or $b$.

A *finite word*, usually denoted by $u$, $v$, or $w$, over an alphabet $\Sigma$ is a function $w : n \to \Sigma$ for some $n$. We call $n$ the *length* of $w$ and write $|w| = n$. The unique word of length 0 is called *empty word* and is written as $\varepsilon$.

Given $\Sigma^n = \{w \mid w \text{ is a word of length } n \text{ over } \Sigma\}$, we define $\Sigma^* = \bigcup\limits_{n \in \mathbb{N}} \Sigma^n$ as the set of all finite words over $\Sigma$.

**Definition 1.2.2.** An *$\omega$-word*, usually denoted by $\alpha$ or $\beta$, over an alphabet $\Sigma$ is a function $\alpha : \omega \to \Sigma$. $\omega$ is the length of $\alpha$ and we write $|\alpha| = \omega$. The set $\Sigma^\omega$ then describes the set of all $\omega$-words over $\Sigma$.

**Definition 1.2.3.** A *language* over an alphabet $\Sigma$ is a set of words $L \subseteq \Sigma^* \cup \Sigma^\omega$. In the context we use it should always be clear whether we are using finite words or $\omega$-words.

**Definition 1.2.4.** Let $v, w \in \Sigma^*$ and $w_i \in \Sigma^*$ for all $i \in \mathbb{N}$ be words over $\Sigma$ and $\alpha \in \Sigma^\omega$ be an $\omega$-word over $\Sigma$.

The *concatenation* of $v$ and $w$ (denoted by $v \cdot w$) is a word $u$ such that:

$$u : |v| + |w| \to \Sigma, i \mapsto \begin{cases} v(i) & \text{if } i < |v| \\ w(i - |v|) & \text{else} \end{cases}$$

The *concatenation* of $w$ and $\alpha$ (denoted by $w \cdot \alpha$) is an $\omega$-word $\beta$ such that:

$$\beta : \mathbb{N} \to \Sigma, i \mapsto \begin{cases} w(i) & \text{if } i < |w| \\ \alpha(i - |w|) & \text{else} \end{cases}$$

For some $n \in \mathbb{N}$, the *$n$-iteration* of $w$ (denoted by $w^n$) is a word $u$ such that:

$$u : |w|^n \to \Sigma, i \mapsto w(i \mod |w|)$$

The *$\omega$-iteration* of $w$ (denoted by $w^\omega$) is an $\omega$-word $\alpha$ such that:

$$\beta : \mathbb{N} \to \Sigma, i \mapsto w(i \mod |w|)$$

For the purpose of easier notation and readability, we write singular symbols as words, i.e. for an $a \in \Sigma$ we write $a$ for the word $w_a : \{0\} \to \Sigma, i \mapsto a$.
We also abbreviate $v \cdot w$ to $vw$ and $w \cdot \alpha$ to $w\alpha$. Further, we use $\alpha \cdot \varepsilon = \alpha$ for $\alpha \in \Sigma^\omega$.

**Definition 1.2.5.** Let $L, K \subseteq \Sigma^*$ be a language and $U \subseteq \Sigma^\omega$ be an $\omega$-language.
The *concatenation* of $L$ and $K$ is $L{\cdot}K = \{u \in \Sigma^* \mid \text{There are } v \in L \text{ and } w \in K \text{ such that } u = v{\cdot}w\}$.
The *concatenation* of $L$ and $U$ is $L{\cdot}U = \{\alpha \in \Sigma^\omega \mid \text{There are } w \in L \text{ and } \beta \in U \text{ such that } \alpha = w{\cdot}\beta\}$.
For some $n \in \mathbb{N}$, the *$n$-iteration* of $L$ is $L^n = \{w \in \Sigma^* \mid \text{There is } v \in L \text{ such that } w = v^n\}$.
The *Kleene closure* of $L$ is $L^* = \bigcup_{n \in \mathbb{N}} L^n$.

**Definition 1.2.6.** Let $w \in \Sigma^* \cup \Sigma^\omega$ be a word. We define a substring or subword of $w$ for some $n \leq m \leq |w|$ as $w[n, m] = w(n) \cdot w(n+1) \cdots w(m-1)$. In the case that $m = |w| = \omega$, it is simply $w[n, m] = w(n) \cdot w(n+1) \cdots$. Note that for $n = m$, we have $w[n, m] = \varepsilon$.

**Definition 1.2.7.** Let $v, w \in \Sigma^* \cup \Sigma^\omega$ be words. We call $v$

- a *prefix* of $w$, if there is an $n \in \mathbb{N}_\omega$ with $v = w[0, n]$.
- a *suffix* of $w$, if there is an $n \in \mathbb{N}_\omega$ with $v = w[n, |w|]$.
- an *infix* of $w$, if there are $n, m \in \mathbb{N}_\omega$ with $v = w[n, m]$.

**Definition 1.2.8.** The *occurrence set* of a word $w \in \Sigma^* \cup \Sigma^\omega$ is the set of symbols which occur at least once in $w$.

$$\mathrm{Occ}(w) = \{a \in \Sigma \mid \text{There is an } n \in |w| \text{ such that } w(n) = a.\}$$

The *infinity set* of a word $w \in \Sigma^\omega$ is the set of symbols which occur infinitely often in $w$.

$$\mathrm{Inf}(w) = \{a \in \Sigma \mid \text{For every } n \in \mathbb{N} \text{ there is a } m > n \text{ such that } w(m) = a.\}$$

## 1.3 Automata

**Definition 1.3.1.** Let $Q$ be a set, $\Sigma$ an alphabet, and $\delta : Q \times \Sigma \to Q$ a function. We call $\mathcal{S} = (Q, \Sigma, \delta)$ a *deterministic transition structure*. We call $Q$ the states or state space.

For $q \in Q$ and a word $w \in \Sigma^* \cup \Sigma^\omega$, we call $\rho \in Q^{1+|w|}$ the *run* of $\mathcal{S}$ on $w$ starting in $q$ if $\rho(0) = q$ and for all $i$, $\rho(i+1) = \delta(\rho(i), w(i))$.

**Definition 1.3.2.** Let $\mathcal{S} = (Q, \Sigma, \delta)$ be a deterministic transition structure. For a set $\Omega \subseteq Q^* \cup Q^\omega$, we say that $\mathcal{S}$ has acceptance condition $\Omega$.

We say that a run $\rho$ of $\mathcal{A}$ on some $w \in \Sigma^*$ is *accepting*, if $\rho \in \Omega$; otherwise, the run is *rejecting*. In either case, we say that $\mathcal{A}$ accepts or rejects $w$. The *language* of $\mathcal{A}$ with $\Omega$ is the set of all words and $\omega$-words that are accepted by $\mathcal{A}$.

**Definition 1.3.3.** A *deterministic finite automaton* (or DFA) is a tuple $\mathcal{A} = (Q, \Sigma, \delta, F)$, where $F \subseteq Q$, such that $(Q, \Sigma, \delta)$ is a deterministic transition structure and has acceptance condition $\Omega = \{\rho \in Q^* \mid \rho(|\rho| + 1) \in F\}$. For the language of $(Q, \Sigma, \delta)$ with $\Omega$, we write $L(\mathcal{A})$.

**Definition 1.3.4.** A *deterministic parity automaton* (or DPA) is a tuple $\mathcal{A} = (Q, \Sigma, \delta, c)$, where $c : Q \to \mathbb{N}$, such that $(Q, \Sigma, \delta)$ is a deterministic transition structure and has acceptance condition $\Omega = \{\rho \in Q^* \mid \min \mathrm{Inf}(c(\rho)) \text{ is even}\}$. For the language of $(Q, \Sigma, \delta)$ with $\Omega$, we write $L(\mathcal{A})$.

We call the DPA a *Büchi automaton* (or DBA) if $c(Q) \subseteq \{0, 1\}$. In that case, we use $F$ instead of $c$.

**Definition 1.3.5.** Let $\mathcal{S} = (Q, \Sigma, \delta)$ be a deterministic transition structure. We define $\delta^* : Q \times \Sigma^* \to Q$ as $\delta^*(q, \varepsilon) = q$ and $\delta^*(q, w \cdot a) = \delta(\delta^*(q, w), a)$.

**Definition 1.3.6.** Let $\mathcal{A} = (Q, \Sigma, \delta, c)$ be a DPA. We define $c^* : Q \times \Sigma^* \to \mathbb{N}$ as $c^*(q, w) = c(\delta^*(q, w))$.

**Definition 1.3.7.** Let $\mathcal{S} = (Q, \Sigma, \delta)$ be a deterministic transition structure and let $R \subseteq Q \times Q$ be an equivalence relation over $Q$. We call $R$ a *congruence relation* if for all $(p, q) \in R$ and all $a \in \Sigma$, also $(\delta(p, a), \delta(q, a)) \in R$.

## 1.4 General Results

We first use this section to establish some general results that are used multiple times in the upcoming proofs.

### 1.4.1 Representative Merge

**Definition 1.4.1.** Let $\mathcal{A} = (Q, \Sigma, \delta, c)$ be a DPA and let $\emptyset \neq C \subseteq M \subseteq Q$. Let $\mathcal{A}' = (Q', \Sigma, \delta', c')$ be another DPA. We call $\mathcal{A}'$ a *representative merge of $\mathcal{A}$ w.r.t. $M$ by candidates $C$* if it satisfies the following:

- There is a state $r_M \in C$ such that $Q' = (Q \setminus M) \cup \{r_M\}$.
- $c' = c \restriction_{Q'}$.
- Let $p \in Q'$ and $\delta(p, a) = q$. If $q \notin M$, then $\delta'(p, a) = q$. If $q \in M$, then $\delta'(p, a) = r_M$.

We call $r_M$ the *representative* of $M$ in the merge. We might omit $C$ and implicitly assume $C = M$.

**Definition 1.4.2.** Let $\mathcal{A} = (Q, \Sigma, \delta, c)$ be a DPA and let $\sim \subseteq Q \times Q$ be an equivalence relation over $Q$. Let $\mathcal{A}' = (Q', \Sigma, \delta', c')$ be another DPA. We call $\mathcal{A}'$ a *representative merge of $\mathcal{A}$ w.r.t. $\sim$* if $\mathcal{A}'$ can be constructed as follows:

Let $n$ be the number of equivalence classes in $\sim$ and let $\kappa_1, \dots, \kappa_n$ be an arbitrary enumeration of them. Let $\mathcal{A}_0 = \mathcal{A}$ and for all $i$, let $\mathcal{A}_{i+1}$ be a representative merge of $\mathcal{A}_i$ w.r.t. $\kappa_{i+1}$. Then $\mathcal{A}' = \mathcal{A}_n$.

If $\sim$ is a congruence relation, then the representative merge will actually be the same as the quotient automaton that is used commonly in state space reduction of automata. The following Lemma formally proofs that this definition actually makes sense, as building representative merges is commutative if the merge sets are disjoint.

**Lemma 1.4.1.** *Let $\mathcal{A} = (Q, \Sigma, \delta, c)$ be a DPA and let $M_1, M_2 \subseteq Q$. Let $\mathcal{A}_1$ be a representative merge of $\mathcal{A}$ w.r.t. $M_1$ by some candidates $C_1$. Let $\mathcal{A}_{12}$ be a representative merge of $\mathcal{A}_1$ w.r.t. $M_2$ by some candidates $C_2$. If $M_1$ and $M_2$ are disjoint, then there is a representative merge $\mathcal{A}_2$ of $\mathcal{A}$ w.r.t. $M_2$ by candidates $C_2$ such that $\mathcal{A}_{12}$ is a representative merge of $\mathcal{A}_2$ w.r.t $M_1$ by candidates $C_1$.*

*Proof.* By choosing the same representative $r_{M_1}$ and $r_{M_2}$ in the merges, this is a simple application of the definition. □

The following Lemma, while simple to prove, is interesting and will find use in multiple proofs of correctness later on.

**Lemma 1.4.2.** *Let $\mathcal{A}$ be a DPA. Let $\sim$ be a congruence relation on $Q$ and let $M \subseteq Q$ such that for all $x, y \in M$, $x \sim y$. Let $\mathcal{A}'$ be a representative merge of $\mathcal{A}$ w.r.t. $M$ by candidates $C$. Let $\rho$ and $\rho'$ be runs of $\mathcal{A}$ and $\mathcal{A}'$ on some $\alpha$. Then for all $i$, $\rho(i) \equiv \rho'(i)$.*

*Proof.* We use a proof by induction. For $i = 0$, we have $\rho(0) = q_0$ for some $q_0 \in Q$ and $\rho'(0) = r_{[q_0]_M}$. By choice of the representative, $q_0 \in M$ and $r_{[q_0]_M} \in M$ and thus $q_0 \sim r_{[q_0]_M}$.

Now consider some $i + 1 > 0$. Then $\rho'(i + 1) = r_{[q]_M}$ for $q = \delta(\rho'(i), \alpha(i))$. By induction we know that $\rho(i) \sim \rho'(i)$ and thus $\delta(\rho(i), \alpha(i)) = \rho(i + 1) \sim q$. Further, we know $q \sim r_{[q]_M}$ by the same argument as before. Together this lets us conclude in $\rho(i + 1) \sim q \sim \rho'(i + 1)$. □

### 1.4.2 Reachability

**Definition 1.4.3.** Let $\mathcal{S} = (Q, \Sigma, \delta)$ be a deterministic transition structure. We define the *reachability order* $\preceq^{\mathcal{S}}_{\text{reach}}$ as $p \preceq^{\mathcal{S}}_{\text{reach}} q$ if and only if $q$ is reachable from $p$.

We want to note here that we always assume for all automata to only have one connected component, i.e. for all states $p$ and $q$, there is a state $r$ such that $p$ and $q$ are both reachable from $r$. In practice, most automata have an predefined initial state and a simple depth first search can be used to eliminate all unreachable states.

**Lemma 1.4.3.** *$\preceq^{\mathcal{S}}_{reach}$ is a preorder.*

**Definition 1.4.4.** Let $\mathcal{S} = (Q, \Sigma, \delta)$ be a deterministic transition structure. We call a relation $\preceq$ a *total extension of reachability* if it is a minimal superset of $\preceq^{\mathcal{S}}_{\text{reach}}$ that is also a total preorder.

For $p \preceq q$ and $q \preceq p$, we write $p \simeq q$.

### 1.4.3 Equivalence Relations

In general, we use the symbol $\equiv$ to denote equivalence relations, mostly between states of an automaton. The following is a comprehensive list of all relevant equivalence relations that we use.

- Language equivalence, $\equiv_L$. Defined below.
- Moore equivalence, $\equiv_M$. Defined below.
- Schewe equivalence, $\equiv_{\text{Sch}}$. Defined in
- Delayed simulation equivalence, $\equiv_{\text{de}}$. Defined in
- Path refinement equivalence, $\equiv_{\text{PR}}$. Defined in
- Threshold Moore equivalence, $\equiv_{\text{TM}}$. Defined in
- Labeled SCC filter equivalence, $\equiv_{\text{LSF}}$. Defined in

**Definition 1.4.5.** Let $\mathcal{A}$ be an $\omega$-automaton with state set $Q$. We define *language equivalence* over $Q$ as $p \equiv_L q$ if and only if for all words $\alpha \in \Sigma^\omega$, $\mathcal{A}$ accepts $\alpha$ from $p$ iff $\mathcal{A}$ accepts $\alpha$ from $q$.

**Lemma 1.4.4.** *$\equiv_L$ is a congruence relation.*

*Proof.* It is obvious that $\equiv_L$ is an equivalence relation. For two states $p \equiv_L q$ and some successors $p' = \delta(p, a)$ and $q' = \delta(q, a)$, it must be true that $p' \equiv_L q'$. Otherwise there is a word $\alpha \in \Sigma^\omega$ that is accepted from $p'$ and rejected from $q'$ (or vice-versa). Then $a \cdot \alpha$ is rejected from $p$ and accepted from $q$ and thus $p \not\equiv_L q$. □

**Definition 1.4.6.** Let $\mathcal{A} = (Q, \Sigma, \delta, c)$ be a DPA. We define *Moore equivalence* over $Q$ as $p \equiv_M q$ if and only if for all words $w \in \Sigma^*$, $c(\delta^*(p, w)) = c(\delta^*(q, w))$.

**Lemma 1.4.5.** *$\equiv_M$ is a congruence relation.*

*Proof.* It is obvious that $\equiv_M$ is an equivalence relation. For two states $p \equiv_L q$ and some successors $p' = \delta(p, a)$ and $q' = \delta(q, a)$, it must be true that $p' \equiv_M q'$. Otherwise there is a word $w \in \Sigma^*$ such that $c(\delta^*(p', w)) \neq c(\delta^*(q', w))$. Then $c(\delta^*(p, aw)) \neq c(\delta^*(q, aw))$ and thus $p \not\equiv_M q$. $\square$

# Bibliography

[1] Filippo Bonchi and Damien Pous. Checking nfa equivalence with bisimulations up to congruence. In *Proceedings of the 40th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '13, pages 457–468, New York, NY, USA, 2013. ACM.

[2] Julius Richard Büchi. On a decision method in restricted second order arithmetic. 1966.

[3] J.-M. Champarnaud and F. Coulon. Nfa reduction algorithms by means of regular inequalities. *Theoretical Computer Science*, 327(3):241 – 253, 2004. Developments in Language Theory.

[4] Kousha Etessami, Thomas Wilke, and Rebecca A. Schuller. Fair simulation relations, parity games, and state space reduction for büchi automata. In *Automata, Languages and Programming*, pages 694–707, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[5] John Hopcroft. An n log n algorithm for minimizing states in a finite automaton. *An N Log N Algorithm for Minimizing States in A Finite Automaton*, page 15, 01 1971.

[6] Tao Jiang and B. Ravikumar. Minimal nfa problems are hard. In *Automata, Languages and Programming*, pages 629–640, Berlin, Heidelberg, 1991. Springer Berlin Heidelberg.

[7] Richard Mayr and Lorenzo Clemente. Advanced automata minimization. In *POPL 2013*, Oct 2012.

[8] Warren S. McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. 1943. *Bulletin of mathematical biology*, 52 1-2:99–115; discussion 73–97, 1990.

[9] Edward F. Moore. Gedanken-experiments on sequential machines. In Claude Shannon and John McCarthy, editors, *Automata Studies*, pages 129–153. Princeton University Press, Princeton, NJ, 1956.

[10] Nir Piterman. From nondeterministic büchi and streett automata to deterministic parity automata. *Logical Methods in Computer Science*, 3(3), 2007.

[11] Michael O Rabin and Dana Scott. Finite automata and their decision problems. *IBM Journal of Research and Development*, 3:114–125, 04 1959.

[12] Sven Schewe. Beyond Hyper-Minimisation—Minimising DBAs and DPAs is NP-Complete. In Kamal Lodaya and Meena Mahajan, editors, *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2010)*, volume 8 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 400–411, Dagstuhl, Germany, 2010. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[13] Wolfgang Thomas. Handbook of theoretical computer science (vol. b). chapter Automata on Infinite Objects, pages 133–191. MIT Press, Cambridge, MA, USA, 1990.