# 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 $x$. We denote the set of equivalence classes by $\mathfrak{C}(R) = \{[x]_R \mid x \in R\}$.

## 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 $L^* = \bigcup\limits_{n \in \mathbb{N}} L^n$.

**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 $w$.

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

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

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

## 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 $\Omega$ from $q \in Q$ is the set of all words and $\omega$-words that are accepted by $\mathcal{A}$ from $q$.

**Definition 1.3.3.** Let $\mathcal{S} = (Q, \Sigma, \delta)$ be a deterministic transition structure. A *strongly connected component*, or SCC, is a set $S \subseteq Q$ such that for all $p, q \in S$, there is a $w \in \Sigma^*$ with $\delta^*(p, w) = q$.

An SCC $S$ is *trivial* if it contains only one state $q$ and $\delta(q, a) \neq q$ for all $a \in \Sigma$.

**Definition 1.3.4.** 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 $\Omega$ from $q$, we write $L(\mathcal{A}, q)$.

**Definition 1.3.5.** 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 $\Omega = \{\rho \in Q^* \mid \min \operatorname{Inf}(c(\rho)) \text{ is even}\}$. For the language of $(Q, \Sigma, \delta)$ with $\Omega$ from $q$, we write $L(\mathcal{A}, q)$.

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 1.3.6.** 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 1.3.7.** Let $\mathcal{A} = (Q, \Sigma, \delta, c)$ be a DPA. We define $c^* : Q \times (\Sigma^* \cup \Sigma^\omega) \to (\mathbb{N}^* \cup \mathbb{N}^\omega)$ as $c^*(q, w) : 1 + |w| \to \mathbb{N}, i \mapsto c(\delta^*(q, w[0, i]))$.

Chapter 2

# Theory

## 2.1 General Results

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

### 2.1.1 Equivalence Relations

In general, we use the symbol $\equiv$ to denote equivalence relations, mostly between states of an automata. In general, we have automata $\mathcal{A}$ and $\mathcal{B}$ with states $p$ and $q$ from there respective state spaces. Our relations are then defined on $(\mathcal{A}, p) \equiv (\mathcal{B}, q)$.

**Definition 2.1.1.** Assuming that $\mathcal{A}$ is a fixed automaton that is obvious in context and $p$ and $q$ are both states in $\mathcal{A}$, we shorten $(\mathcal{A}, p) \equiv (\mathcal{A}, q)$ to $p \equiv q$.

Furthermore, we write $\mathcal{A} \equiv \mathcal{B}$ if for every $p$ in $\mathcal{A}$ there is a $q$ in $\mathcal{B}$ such that $(\mathcal{A}, p) \equiv (\mathcal{B}, q)$; and the same holds with $\mathcal{A}$ and $\mathcal{B}$ exchanged.

**Definition 2.1.2.** Let $\mathcal{A} = (Q_1, \Sigma, \delta_1)$ and $\mathcal{B} = (Q_2, \Sigma, \delta_2)$ be deterministic transition structures and let $\sim \subseteq (\{\mathcal{A}\} \times Q_1) \times (\{\mathcal{B}\} \times Q_2)$ be an equivalence relation. We call $R$ a *congruence relation* if for all $(\mathcal{A}, p) \sim (\mathcal{B}, q)$ and all $a \in \Sigma$, also $(\mathcal{A}, \delta_1(p, a)) \sim (\mathcal{B}, \delta_2(q, a))$.

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.
- Priority almost equivalence, $\equiv_\dagger$. Defined below.
- 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

Immediately we define the three first of these relations and show that they are computable.

#### Language Equivalence

**Definition 2.1.3.** Let $\mathcal{A}$ and $\mathcal{B}$ be $\omega$-automata. We define *language equivalence* as $(\mathcal{A}, p) \equiv_L (\mathcal{B}, q)$ if and only if for all words $\alpha \in \Sigma^\omega$, $\mathcal{A}$ accepts $\alpha$ from $p$ iff $\mathcal{B}$ accepts $\alpha$ from $q$.

**Lemma 2.1.1.** *$\equiv_L$ is a congruence relation.*

*Proof.* It is obvious that $\equiv_L$ is an equivalence relation. For two states $(\mathcal{A}, p) \equiv_L (\mathcal{B}, q)$ and some successors $p' = \delta_1(p, a)$ and $q' = \delta_2(q, a)$, it must be true that $(\mathcal{A}, p') \equiv_L (\mathcal{B}, q')$. 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 $p \not\equiv_L q$. $\square$

**Lemma 2.1.2.** *Language equivalence of a given DPA can be computed in $\mathcal{O}(|Q|^2 \cdot |c(Q)|^2)$.*

*Proof.* The algorithm is based partially on [2].

Let $\mathcal{A} = (Q, \Sigma, \delta, c)$ be the DPA that we want to compute $\equiv_L$ on. We construct a labeled deterministic transition structure $\mathcal{B} = (Q \times Q, \Sigma, \delta', d)$ with $\delta'((p_1, p_2), a) = (\delta(p_1, a), \delta(p_2, a))$ and $d((p_1, p_2)) = (c(p_1), c(p_2)) \in \mathbb{N}^2$. Then, for every $i, j \in c(Q)$, let $\mathcal{B}_{i,j} = \mathcal{B} \restriction_{Q_{i,j}}$ with $Q_{i,j} = \{(p_1, p_2) \in Q \times Q \mid c(p_1) \geq i, c(p_2) \geq j\}$, i.e. remove all states which have first priority less than $i$ or second priority less than $j$.

For each $i$ and $j$, let $S_{i,j} \subseteq 2^{Q \times Q}$ be the set of all SCCs in $\mathcal{B}_{i,j}$ and let $S = \bigcup_{i,j} S_{i,j}$. From this set $S$, remove all SCCs $s \subseteq Q \times Q$ in which the parity of the smallest priority in the first component differs from the parity of the smallest priority in the second component. The "filtered" set we call $S'$. For any two states $p, q \in Q$, $p \not\equiv_L q$ iff there is a pair $(p', q') \in \bigcup S'$ that is reachable from $(p, q)$ in $\mathcal{B}$.

We omit the correctness proof of the algorithm here. Regarding the runtime, observe that $\mathcal{B}$ has size $\mathcal{O}(|Q|^2)$ and we create $\mathcal{O}(|c(Q)|^2)$ copies of it. All other steps like computing the SCCs can then be done in linear time in the size of the automata, which brings the total to $\mathcal{O}(|Q|^2 \cdot |c(Q)|^2)$. $\square$

## Priority Almost Equivalence

**Definition 2.1.4.** Let $\mathcal{A} = (Q_1, \Sigma, \delta_1, c_1)$ and $\mathcal{B} = (Q_2, \Sigma, \delta_2, c_2)$ be DPAs. We define *priority almost equivalence* as $(\mathcal{A}, p) \equiv_\dagger (\mathcal{B}, q)$ if and only if for all words $\alpha \in \Sigma^\omega$, $c_1^*(p, \alpha)$ and $c_2^*(q, \alpha)$ differ at only finitely many positions.

**Lemma 2.1.3.** *Priority almost equivalence is a congruence relation.*

*Proof.* It is obvious that $\equiv_\dagger$ is an equivalence relation. For two states $(\mathcal{A}, p) \equiv_\dagger (\mathcal{B}, q)$ and some successors $p' = \delta(p, a)$ and $q' = \delta(q, a)$, it must be true that $(\mathcal{A}, p') \equiv_\dagger (\mathcal{B}, q')$. Otherwise there is a word $\alpha \in \Sigma^\omega$ such that $c_1^*(p', \alpha)$ and $c_2^*(q', \alpha)$ differ at infinitely many positions. Then $c_1^*(p, a\alpha)$ and $c_2^*(q, a\alpha)$ also differ at infinitely many positions and thus $(\mathcal{A}, p) \not\equiv_\dagger (\mathcal{B}, q)$. $\square$

The following definition is used as an intermediate step on the way to computing $\equiv_\dagger$.

**Definition 2.1.5.** Let $\mathcal{A} = (Q_1, \Sigma, \delta_1, c_1)$ and $\mathcal{B} = (Q_2, \Sigma, \delta_2, c_2)$ be DPAs. We define the deterministic Büchi automaton $\mathcal{A} \top \mathcal{B} = (Q_1 \times Q_2, \Sigma, \delta_\top, F_\top)$ with $\delta_\top((q_1, q_2), a) = (\delta_1(q_1, a), \delta_2(q_2, a))$. The transition structure is a common product automaton.

The final states are $F_\top = \{(p, q) \in Q_1 \times Q_2 \mid c_1(p) \neq c_2(q)\}$, i.e. every pair of states at which the priorities differ.

**Lemma 2.1.4.** *$\mathcal{A} \top \mathcal{B}$ can be computed in time $\mathcal{O}(|\mathcal{A}| \cdot |\mathcal{B}|)$.*

*Proof.* The definition already provides a rather straightforward description of how to compute $\mathcal{A} \top \mathcal{B}$. Each state only requires constant time (assuming that $\delta$ and $c$ can be evaluated in such) and has $|\mathcal{A}| \cdot |\mathcal{B}|$ many states. $\square$

**Lemma 2.1.5.** *Let $\mathcal{A} = (Q_1, \Sigma, \delta_1, c_1)$ and $\mathcal{B} = (Q_2, \Sigma, \delta_2, c_2)$ be DPAs. $(\mathcal{A}, p) \equiv_\dagger (\mathcal{B}, q)$ iff $L(\mathcal{A} \top \mathcal{B}, (p, q)) = \emptyset$.*

*Proof.* For the first direction of implication, let $L(\mathcal{A} \intercal \mathcal{B}, (p_0, q_0)) \neq \emptyset$, so there is a word $\alpha$ accepted by that automaton. Let $(p, q)(p_1, q_1)(p_2, q_2)\cdots$ be the accepting run on $\alpha$. Then $pp_1 \cdots$ and $qq_1 \cdots$ are the runs of $\mathcal{A}$ and $\mathcal{B}$ on $\alpha$ respectively. Whenever $(p_i, q_i) \in F_\intercal$, $p_i$ and $q_i$ have different priorities. As the run of the product automaton vists infinitely many accepting states, $\alpha$ is a witness for $p$ and $q$ being not priority almost-equivalent.

For the second direction, let $p$ and $q$ be not priority almost-equivalent, so there is a witness $\alpha$ at which infinitely many positions differ in priority. Analogously to the first direction, this means that the run of $\mathcal{A} \intercal \mathcal{B}$ on the same word is accepting and therefore the language is not empty. □

**Corollary 2.1.6.** *Priority almost equivalence of a given DPA can be computed in quadratic time.*

*Proof.* By Lemma 2.1.4, we can compute $\mathcal{A} \intercal \mathcal{A}$ in quadratic time. The emptiness problem for deterministic Büchi automata is solvable in linear time by checking reachability of loops that contain a state in $F$. □

## Moore Equivalence

**Definition 2.1.6.** Let $\mathcal{A} = (Q_1, \Sigma, \delta_1, c_1)$ and $\mathcal{B} = (Q_2, \Sigma, \delta_2, c_2)$ be DPAs. We define *Moore equivalence* as $(\mathcal{A}, p) \equiv_M (\mathcal{B}, q)$ if and only if for all words $w \in \Sigma^*$, $c_1(\delta^*(p, w)) = c_2(\delta^*(q, w))$.

**Lemma 2.1.7.** $\equiv_M$ *is a congruence relation.*

*Proof.* It is obvious that $\equiv_M$ is an equivalence relation. For two states $(\mathcal{A}, p) \equiv_M (\mathcal{B}, q)$ and some successors $p' = \delta(p, a)$ and $q' = \delta(q, a)$, it must be true that $(\mathcal{A}, p') \equiv_M (\mathcal{B}, q')$. Otherwise there is a word $w \in \Sigma^*$ such that $c_1(\delta_1^*(p', w)) \neq c_2(\delta_2^*(q', w))$. Then $c_1(\delta_1^*(p, aw)) \neq c_2(\delta_2^*(q, aw))$ and thus $(\mathcal{A}, p) \not\equiv_M (\mathcal{B}, q)$. □

**Lemma 2.1.8.** *Moore equivalence of a given DPA can be computed in log-linear time.*

*Proof.* We refer to [3]. The given algorithm can be adapted to Moore automata without changing the complexity. □

**Lemma 2.1.9.** $\equiv_M \subseteq \equiv_\dagger \subseteq \equiv_L$

*Proof.* Let $\mathcal{A} = (Q_\mathcal{A}, \Sigma, q_0^\mathcal{A}, \delta_\mathcal{A}, c_\mathcal{A})$ and $\mathcal{B} = (Q_\mathcal{B}, \Sigma, q_0^\mathcal{B}, \delta_\mathcal{B}, c_\mathcal{B})$ be two DPA that are priority almost-equivalent and assume towards a contradiction that they are not language equivalent. Due to symmetry we can assume that there is a $w \in L(\mathcal{A}) \setminus L(\mathcal{B})$.

Consider $\alpha = \lambda_\mathcal{A}(q_0^\mathcal{A}, w)$ and $\beta = \lambda_\mathcal{B}(q_0^\mathcal{B}, w)$, the priority outputs of the automata on $w$. By choice of $w$, we know that $a := \max \mathrm{Inf}(\alpha)$ is even and $b := \max \mathrm{Inf}(\beta)$ is odd. Without loss of generality, assume $a > b$. That means $a$ is seen only finitely often in $\beta$ but infinitely often in $a$. Hence, $\alpha$ and $\beta$ differ at infinitely many positions where $a$ occurs in $\alpha$. That would mean $w$ is a witness that the two automata are not priority almost-equivalent, contradicting our assumption. □

### 2.1.2 Representative Merge

**Definition 2.1.7.** 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 $q \in M$, then $\delta'(p, a) = r_M$. Otherwise, $\delta'(p, a) = q$.

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

**Definition 2.1.8.** Let $\mathcal{A} = (Q, \Sigma, \delta, c)$ be a DPA and let $\mu : D \to (2^Q \setminus \emptyset)$ be a function for some $D \subseteq 2^Q$. If all sets in $D$ are pairwise disjoint and for all $X \in D$, $\mu(X) \subseteq X$, we call $\mu$ a *merger function*.

A DPA $\mathcal{A}'$ is a representative merge of $\mathcal{A}$ w.r.t. $\mu$ if there is an enumeration $X_1, \ldots, X_{|D|}$ of $D$ and a sequence of automata $\mathcal{A}_0, \ldots, \mathcal{A}_{|D|}$ such that $\mathcal{A}_0 = \mathcal{A}$, $\mathcal{A}_{|D|} = \mathcal{A}'$ and every $\mathcal{A}_{i+1}$ is a representative merge of $\mathcal{A}_i$ w.r.t. $X_{i+1}$ by candidates $\mu(X_{i+1})$.

The following Lemma formally proofs that this definition actually makes sense, as building representative merges is commutative if the merge sets are disjoint.

**Lemma 2.1.10.** *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 2.1.11.** *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 $\rho(i + 1) \sim q \sim \rho'(i + 1)$. □

The following is a comprehensive list of all relevant merger functions that we use.

- Quotient merger, $\mu_{\tilde{\div}}$. Defined below.

- Moore merger, $\mu_M$. Defined below.
- Skip merger, $\mu^{\sim}_{\text{skip}}$. Defined in section **??**.

**Quotient merger**

**Definition 2.1.9.** Let $\sim$ be a congruence relation. We define the *quotient merger* $\mu^{\sim}_{\div} : \mathfrak{C}(\sim) \to 2^Q, \kappa \mapsto \kappa$.

**Lemma 2.1.12.** *Let $\mathcal{A} = (Q, \Sigma, \delta, c)$ be a DPA and let $\sim$ be a congruence relation such that $p \sim q$ implies $c(p) = c(q)$. Then $\mathcal{A}$ is Moore equivalent to every representative merge w.r.t. $\mu^{\sim}_{\div}$.*

*Proof.* Let $\mathcal{A}' = (Q', \Sigma, \delta', c')$ be a representative merge. For every $q \in Q$, we prove that $(\mathcal{A}, q) \equiv_M (\mathcal{A}', r_{[q]_\sim})$. Since all states in $\mathcal{A}'$ are representatives of that form and every representative exists in $\mathcal{A}$ as well, this suffices to prove Moore equivalence.

Let $\alpha \in \Sigma^\omega$ be a word and let $\rho$ and $\rho'$ be the runs of $\mathcal{A}$ and $\mathcal{A}'$ on $\alpha$ starting in $q$ and $r_{[q]_\sim}$. For every $i \in \mathbb{N}$, we have $\rho'(i) = [\rho(i)]_\sim$ and thus $c'(\rho'(i)) = c(\rho(i))$. Therefore, $\rho$ is accepting iff $\rho'$ is accepting. □

**Definition 2.1.10.** We define the special *Moore merger* $\mu_M = \mu^{\equiv_M}_{\div}$.

### 2.1.3 Reachability

**Definition 2.1.11.** 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 2.1.13.** *$\preceq^{\mathcal{S}}_{\text{reach}}$ is a preorder.*

**Definition 2.1.12.** 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 $p \simeq q$.

**Lemma 2.1.14.** *For a given deterministic transition structure $\mathcal{S}$, a total extension of reachability is computable in $\mathcal{O}(|\mathcal{S}|)$.*

*Proof.* Using e.g. Kosaraju's algorithm **??**, the SCCs of $\mathcal{A}$ can be computed in linear time. We can now build a DAG from $\mathcal{A}$ by merging all states in an SCC into a single state; iterate over all transitions $(p, a, q)$ and add an $a$-transition from the merged representative of $p$ to that of $q$. Assuming efficient data structures for the computed SCCs, this DAG can be computed in $O(|\mathcal{A}|)$ time.

To finish the computation of $\preceq$, we look for a topological order on that DAG. This is a total preorder on the SCCs that is compatible with reachability. All that is left to be done is to extend that order to all states. □

## 2.2 Fritz & Wilke

### 2.2.1 Delayed Simulation Game

In this section we consider delayed simulation games and variants thereof on DPAs. This approach is based on the paper [1], which considered the games for alternating parity automata. The DPAs we use are a special case of these APAs and therefore worth examining.

**Definition 2.2.1.** For convenience, we define two orders for this chapter. First, we introduce ✓ as an "infinity" to the natural numbers and define the **obligation order** $\leq_{\checkmark} \subseteq (\mathbb{N} \cup \{\checkmark\}) \times (\mathbb{N} \cup \{\checkmark\})$ as $0 \leq_{\checkmark} 1 \leq_{\checkmark} 2 \leq_{\checkmark} \cdots \leq_{\checkmark} \checkmark$.

Second, we define an order of "goodness" on parity priorities $\preceq_{\mathrm{p}} \subseteq \mathbb{N} \times \mathbb{N}$ as $0 \preceq_{\mathrm{p}} 2 \preceq_{\mathrm{p}} 4 \preceq_{\mathrm{p}} \cdots \preceq_{\mathrm{p}} 5 \preceq_{\mathrm{p}} 3 \preceq_{\mathrm{p}} 1$.

**Definition 2.2.2.** Let $\mathcal{A} = (Q, \Sigma, \delta, c)$ be a DPA. We define the *delayed simulation automaton* $\mathcal{A}_{\mathrm{de}}(p, q) = (Q_{\mathrm{de}}, \Sigma, \delta_{\mathrm{de}}, F_{\mathrm{de}})$ with starting state $q_0^{\mathrm{de}}(p, q) = (p, q, \gamma(c(p), c(q), \checkmark))$, which is a deterministic Büchi automaton, as follows.

- $Q_{\mathrm{de}} = Q \times Q \times (\mathrm{img}(c) \cup \{\checkmark\})$, i.e. the states are given as triples in which the first two components are states from $\mathcal{A}$ and the third component is either a priority from $\mathcal{A}$ or ✓.
- The alphabet remains $\Sigma$.
- $\delta_{\mathrm{de}}((p, q, k), a) = (p', q', \gamma(c(p'), c(q'), k)$, where $p' = \delta(p, a)$, $q' = \delta(q, a)$, and $\gamma$ is the same function as used in the initial state. The first two components behave like a regular product automaton.
- $F_{\mathrm{de}} = Q \times Q \times \{\checkmark\}$.
- The starting state is a triple $(p, q, \gamma(c(p), c(q), \checkmark))$, where $p, q \in Q$ are parameters given to the automaton, and $\gamma$ is defined below.

$\gamma : \mathbb{N} \times \mathbb{N} \times (\mathbb{N} \cup \{\checkmark\}) \to \mathbb{N} \cup \{\checkmark\}$ is the update function of the third component and defines the "obligations" as they are called in []. It is defined as

$$\gamma(i, j, k) = \begin{cases} \checkmark & \text{if } i \text{ is odd and } i \leq_{\checkmark} k \text{ and } j \preceq_{\mathrm{p}} i \\ \checkmark & \text{if } j \text{ is even and } j \leq_{\checkmark} k \text{ and } j \preceq_{\mathrm{p}} i \\ \min_{\leq_{\checkmark}}\{i, j, k\} & \text{else} \end{cases}$$

**Definition 2.2.3.** Let $\mathcal{A}$ be a DPA and let $\mathcal{A}_{\mathrm{de}}$ be the delayed simulation automaton of $\mathcal{A}$. We say that a state $p$ *de*-simulates a state $q$ if $L(\mathcal{A}_{\mathrm{de}}(p, q), q_0^{\mathrm{de}}(p, q)) = \Sigma^\omega$. In that case we write $p \leq_{\mathrm{de}} q$. If also $q \leq_{\mathrm{de}} p$ holds, we write $p \equiv_{\mathrm{de}} q$.

**$\equiv_{\mathrm{de}}$ is a congruence relation.**

Our overall goal is to use $\equiv_{\mathrm{de}}$ to build a quotient automaton of our original DPA. The first step towards this goal is to show that the result is actually a well-defined DPA, by proving that the relation is a congruence.

**Lemma 2.2.1.** *$\gamma$ is monotonous in the third component, i.e. if $k \leq_{\checkmark} k'$, then $\gamma(i, j, k) \leq_{\checkmark} \gamma(i, j, k')$ for all $i, j \in \mathbb{N}$.*

*Proof.* We consider each case in the definition of $\gamma$. If $i$ is odd, $i \leq_{\checkmark} k$ and $j \preceq_{\mathrm{p}} i$, then also $i \leq_{\checkmark} k'$ and $\gamma(i, j, k) = \gamma(i, j, k') = \checkmark$.

If $j$ is even, $j \leq_{\checkmark} k$ and $j \preceq_{\mathrm{p}} i$, then also $j \leq_{\checkmark} k'$ and $\gamma(i, j, k) = \gamma(i, j, k') = \checkmark$.

Otherwise, $\gamma(i, j, k) = \min\{i, j, k\}$ and $\gamma(i, j, k') = \min\{i, j, k'\}$. Since $k \leq_{\checkmark} k'$, $\gamma(i, j, k) \leq_{\checkmark} \gamma(i, j, k')$. $\square$

**Lemma 2.2.2.** *Let $\mathcal{A}$ be a DPA and let $p, q \in Q$, $k \in \mathbb{N} \cup \{\checkmark\}$. If the run of $\mathcal{A}_{\mathrm{de}}$ starting at $(p, q, k)$ on some $\alpha \in \Sigma^\omega$ is accepting, then for all $k \leq_{\checkmark} k'$ also the run of $\mathcal{A}_{\mathrm{de}}$ starting at $(p, q, k')$ on $\alpha$ is accepting.*

*Proof.* Let $\rho$ be the run starting at $(p, q, k)$ and let $\rho'$ be the run starting at $(p, q, k')$. Further, let $p_i$, $q_i$, $k_i$, and $k'_i$ be the components of the states of those runs in the $i$-th step. Via induction we show that $k_i \leq_{\checkmark} k'_i$ for all $i$. Since $k_i$ is $\checkmark$ infinitely often, the same must be true for $k'_i$ and $\rho'$ is accepting.

For $i = 0$, we have $k_0 = k \leq_{\checkmark} k' = k'_0$. Otherwise, we have $k_{i+1} = \gamma(c(p_{i+1}), c(q_{i+1}), k_i)$ and $k'_{i+1}$ analogously. The rest follows from Lemma 2.2.1. $\square$

**Lemma 2.2.3.** *Let $\mathcal{A}$ be a DPA and $\rho \in Q_{\mathrm{de}}^\omega$ be a run of $\mathcal{A}_{\mathrm{de}}$ on some word. Let $k \in (\mathbb{N} \cup \{\checkmark\})^\omega$ be the third component during $\rho$. For all $i$, $k(i + 1) \leq_{\checkmark} k(i)$ or $k(i + 1) = \checkmark$.*

*Proof.* Follows directly from the definition of $\gamma$. $\square$

**Lemma 2.2.4.** *Let $\mathcal{A}$ be a DPA with states $p, q \in Q$. For a word $\alpha \in \Sigma^\omega$, let $\rho : i \mapsto (p_i, q_i, k_i)$ be the run of $\mathcal{A}_{\mathrm{de}}(p, q)$ on $\alpha$. If $\rho$ is not accepting, there is a position $n$ such that*

- *$\mathit{Occ}(\{p_i \mid i \geq n\}) = \mathit{Inf}(\{p_i \mid i \in \mathbb{N}\})$,*
- *$\mathit{Occ}(\{q_i \mid i \geq n\}) = \mathit{Inf}(\{q_i \mid i \in \mathbb{N}\})$,*
- *For all $i \geq j \geq n$, $k_i = k_j$ and $k_i \neq \checkmark$.*

*In other words, from $n$ on, $p$ and $q$ only see states that are seen infinitely often, and the obligations of $\rho$ do not change anymore.*

*Proof.* The first two requirements are clear. Since states not in $\mathrm{Inf}(\{p_i \mid i \in \mathbb{N}\})$ only occur finitely often, there must be positions $n_p$ and $n_q$ from which on they do not occur anymore at all.

For the third requirement, we know that from some point on, the obligations $k$ never become $\checkmark$ anymore, as the run would be accepting otherwise. By Lemma 2.2.3, $k$ can only become lower from there on. As $\leq_{\checkmark}$ is a well-ordering, a minimum must be reached at some point $n_k$.

The position $n_0 = \max\{n_p, n_q, n_k\}$ satisfies the statement. $\square$

**Lemma 2.2.5.** *Let $\mathcal{A}$ be a DPA with two states $p, q \in Q$. Let $\alpha \in \Sigma^\omega$ be an $\omega$-word and let $\rho_p$ and $\rho_q$ be the respective runs of $\mathcal{A}$ on $\alpha$ starting in $p$ and $q$. If $\min \mathit{Inf}(c(\rho_q)) \preceq_p \min \mathit{Inf}(c(\rho_p))$, then $\alpha \in L(\mathcal{A}_{\mathrm{de}}(p, q))$.*

*Proof.* We write $l_q = \min \mathrm{Inf}(c(\rho_q))$ and $l_p = \min \mathrm{Inf}(c(\rho_p))$. Assume that the Lemma is false, so $l_q \preceq_p l_p$ but $\alpha \notin L(\mathcal{A}_{\mathrm{de}}(p, q))$. Let $k_{13} \in (\mathbb{N} \cup \{\checkmark\})^\omega$ be the third component of the run of $\mathcal{A}_{\mathrm{de}}(p, q)$ on $\alpha$. Let $n_0$ be a position as described in Lemma 2.2.4 (for $\rho$).

**Case 1: $l_q$ is even and $l_q \leq l_p$** We know $k_{13}(n_0) = l_q$, as that is the smaller value. Let $m > n_0$ be a position with $c(\rho_q(m)) = l_q$. Then $c(\rho_q(m))$ is even and $c(\rho_q(m)) = l_q \leq l_p \leq c(\rho_p(m))$, so $c(\rho_q(m)) \preceq_p c(\rho_p(m))$. Also we have $c(\rho_q(m)) \leq k_{13}(m-1) = l_q$ and therefore $k_{13}(m) = \checkmark$, which contradicts the choice of $n_0$.

**Case 2: $l_p$ is odd and $l_q \geq l_p$** We know $k_{13}(n_0) = l_p$. Let $m > n_0$ be a position with $c(\rho_p(m)) = l_p$. Then $c(\rho_p(m))$ is odd and $c(\rho_p(m)) = l_p \leq l_q \leq c(\rho_q(m))$, so $c(\rho_q(m)) \preceq_p c(\rho_p(m))$. By the same argumentation as above, we deduce $k_{13}(m) = \checkmark$. $\square$

**Lemma 2.2.6.** *Let $\mathcal{A}$ be a DPA. Then $\leq_{\mathrm{de}}$ is reflexive and transitive.*

*Proof.* For reflexivitiy, we need to show that $q \leq_{\mathrm{de}} q$ for all states $q$. This is rather easy to see. For a word $\alpha \in \Sigma^\omega$, the third component of states in the run of $\mathcal{A}_{\mathrm{de}}(q, q)$ on $\alpha$ is always $\checkmark$, as $\gamma(i, i, \checkmark) = \checkmark$.

For transitivity, let $q_1 \leq_{\mathrm{de}} q_2$ and $q_2 \leq_{\mathrm{de}} q_3$. Assume towards a contradiction that $q_1 \not\leq_{\mathrm{de}} q_3$, so there is a word $\alpha \notin L(\mathcal{A}_{\mathrm{de}}(q_1, q_3))$. We consider the three runs $\rho_{12}$, $\rho_{23}$, and $\rho_{13}$ of $\mathcal{A}_{\mathrm{de}}(q_1, q_2)$, $\mathcal{A}_{\mathrm{de}}(q_2, q_3)$, and $\mathcal{A}_{\mathrm{de}}(q_1, q_3)$ respectively on $\alpha$. Then $\rho_{12}$ and $\rho_{23}$ are accepting, whereas $\rho_{13}$ is not.

Moreover, we use the notation $q_1(i), q_2(i), q_3(i)$ for the states of the run and $k_{12}(i), k_{23}(i), k_{13}(i)$ for the obligations. More specifically for a run $\rho_{ij}$, it is true that $\rho_{ij}(n) = (q_i(n), q_j(n), k_{ij}(n))$.

Let $n_0$ be a position as described in Lemma 2.2.4 (for $\rho_{13}$) and let $l_j = \min\{c(q_j(i)) \mid i \geq n_0\}$ be the lowest priority that $q_j$ reaches after $n_0$. This is equivalent to $l_j = \min \mathrm{Inf}(\{c(q_j(i)) \mid i \in \mathbb{N}\})$. We now show that $l_3 \preceq_p l_1$. By Lemma 2.2.5 this gives us $\alpha \in L(\mathcal{A}_{\mathrm{de}}(q_1, q_3))$, letting us conclude in a contradiction.

**Case 1: $l_2$ is even.** We claim that $l_3$ is even and $l_3 \leq l_2$.

First, to show $l_3 \leq l_2$, let $m \geq n_0$ be a position with $c(q_2(m)) = l_2$ and let $n \geq m$ be the minimal position with $k_{23}(n) = \checkmark$. If $m = n$, then $c(q_3(n)) \preceq_p c(q_2(n)) = l_2$ and therefore $c(q_3(n)) \leq l_2$. Otherwise, from $m$ to $n-1$, $k_{23}$ only grows smaller and is at most $l_2$ (Lemma 2.2.3). As the priority of $q_2$ never becomes an odd number smaller than $l_2$, the only way for $k_{23}(m)$ to be $\checkmark$ is that $c(q_3(m))$ is even and $c(q_3(m)) \leq k_{23}(m-1) \leq l_2$.

Second, assume that $l_3$ is odd and let $m$ be a position with $c(q_3(m)) = l_3$. As $l_2$ is even, we have $k_{23}(m) \leq l_3 < l_2$. At no future position can $c(q_3)$ both be even and smaller than $k_{23}$, so $k_{23}$ never becomes $\checkmark$ again. Thus, $\rho_{23}$ is not accepting.

We claim that $l_1$ is odd or $l_1 \geq l_2$.

Towards a contradiction assume the opposite, so $l_1 < l_2$ and $l_1$ is even. Let $m \geq n_0$ be a position with $c(q_1(m)) = l_1$. Then $c(q_2(m)) \not\preceq_p c(q_1(m))$ and therefore $k_{12}(m) = l_1$. At no position after $m$ can it happen that the conditions for $k_{12}$ to become $\checkmark$ again are satisfied. Thus, $\rho_{12}$ would not be accepting.

If $l_1$ is odd and $l_3$ is even, $l_3 \preceq_p l_1$ follows. For the other case, $l_1$ and $l_3$ both being even with $l_3 \leq l_2 \leq l_1$, that also holds.

**Case 2: $l_2$ is odd.** We skip the details of this case as it works symmetrically to case 1. In particular, we first show that $l_1$ is odd and $l_1 \leq l_2$. We continue with $l_3$ being even or $l_3 \geq l_2$. From these two statements, $l_3 \preceq_p l_1$ again follows. □

**Theorem 2.2.7.** *Let $\mathcal{A}$ be a DPA. Then $\equiv_{\text{de}}$ is a congruence relation.*

*Proof.* The three properties that are required for $\equiv_{\text{de}}$ to be a equivalence relation are rather easy to see. Reflexivity and transitivity have been shown for $\leq_{\text{de}}$ already and symmetry follows from the definition. Congruence requires more elaboration.

Let $p \equiv_{\text{de}} q$ be two equivalent states. Let $a \in \Sigma$ and $p' = \delta(p, a)$ and $q' = \delta(q, a)$. We have to show that also $p' \equiv_{\text{de}} q'$. Towards a contradiction, assume that $p' \not\leq_{\text{de}} q'$, so there is a word $\alpha \notin L(\mathcal{A}_{\text{de}}(p', q'))$. Let $(p', q', k) = \delta_{\text{de}}((p, q, \checkmark), a)$. By Lemma 2.2.2, the run of $\mathcal{A}_{\text{de}}$ on $\alpha$ from $(p', q', k)$ cannot be accepting; otherwise, the run of $\mathcal{A}_{\text{de}}$ from $(p', q', \checkmark)$ would be accepting and $\alpha \in L(\mathcal{A}_{\text{de}}(p', q'))$. Hence, $a\alpha \notin L(\mathcal{A}_{\text{de}}(p, q))$, which means that $p \not\equiv_{\text{de}} q$. □

We want to mention here that $\equiv_{\text{de}}$ is actually an equivalence relation on APAs as well, as was shown in the original paper. However, congruence is the key point at which deterministic automata diverge. Congruence requires something to be true for *all* successors of a state; delayed simulation only requires there to be *one* equivalent pair of successors. Only in deterministic automata is it that these two coincide.

### Alternative characterization

Having described several properties of $\leq_{\text{de}}$ and $\equiv_{\text{de}}$ in the previous part, we briefly want to give an alternative description in corollary 2.2.9 of what it means for two states to be de-simulation equivalent. The intention is to give a more intuitive understanding as well as provide a framework to make future proofs more comfortable.

**Lemma 2.2.8.** *Let $\mathcal{A}$ be a DPA with states $p, q \in Q$. Then $p \preceq_{\text{de}} q$ if and only if the following property holds for all $w \in \Sigma^*$:*

*Let $p' = \delta^*(p, w)$ and $q' = \delta^*(q, w)$. If $c(p')$ is even and $c(p') < c(q')$, then every path from $q'$ eventually reaches a priority at most $c(p')$. On the other hand, if $c(q')$ is odd and $c(q') < c(p')$, then every path from $p'$ eventually reaches a priority at most $c(q')$.*

*Proof.* **If** We show the contrapositive. Let $p \not\preceq_{\text{de}} q$, so there is a word $\alpha \notin L(\mathcal{A}_{\text{de}}(p, q))$ with the run of $\mathcal{A}_{\text{de}}(p, q)$ being $(p_0, q_0, k_0)(p_1, q_1, k_1) \ldots$. By Lemma 2.2.4, there is a position $n$ such that $k_i$ does not change anymore for $i \geq n$. We define $w = \alpha[0, n_0]$ and $\beta = \alpha[n_0 + 1, ]$ (so $\alpha = w\beta$) and claim that $w$ is a valid counterexample for the right-side property.

The first case is: $c(p') < c(q')$ and $c(p')$ is even.
Reading $\beta$ from $q'$ induces a run that never visits a priority less or equal to $c(p')$: Let $u \sqsubset \beta$ and assume that $c(\delta^*(q', u)) \leq c(p')$. By choice of $n$, we know that $k_{|wu|} = k_{|wu|+1} \neq \checkmark$. This can only happen if the "else" case of $\gamma$ is hit, meaning that $k_{|wu|+1} = \min\{k_{|wu|}, c(\delta^*(p', u)), c(\delta^*(q', u))\}$.

Specifically, $c(\delta^*(q', u)) \geq k_{|wu|}$. By choice of $w$ we also have $k_{|wu|} = k_{|w|} \leq c(p')$, so $c(\delta^*(q', u)) = c(p')$.

This, however, means that $c(\delta^*(q', u))$ is even, $c(\delta^*(q', u)) \leq_{\checkmark} k_{|wu|}$, and $c(\delta^*(q', u)) \preceq_p c(p')$ and thus $k_{|wu|+1} = \checkmark$ which is a contradiction.

The second case, $c(q') < c(p')$ and $c(q')$ is odd, works almost identically so we omit the proof here.

**Only If** Again we show the contrapositive: There is a $w \in \Sigma^*$ such that the right-side property is violated. Let this $w$ now be chosen among all these words such that $\min\{c(p'), c(q')\}$ becomes minimal. We now show that $p \not\preceq_{\text{de}} q$.

The first case is: $c(p') < c(q')$ and $c(p')$ is even.
Let $\beta \in \Sigma^\omega$ be a word such that the respective run from $q'$ only sees priorities strictly greater than $c(p')$. Let $(p_0, q_0, k_0)(p_1, q_1, k_1)\dots$ be the run of $\mathcal{A}_{\text{de}}(p, q)$ on $\alpha = w\beta$. We claim that $k_i \neq \checkmark$ for all $i > |w|$. If that is true, then the run is rejecting and $\alpha \notin L(\mathcal{A}_{\text{de}}(p, q))$.

Assume towards a contradiction that $k_i$ does become $\checkmark$ again at some point. Let $j \geq |w|$ be the minimal position with $k_{j+1} = \checkmark$. Then by definition of $\gamma$, $c(q_{j+1}) \leq k_j$ is even or $c(p_{j+1}) \leq k_j$ is odd. In the former case, we would have a contradiction to the choice of $\beta$. In the latter case, we would have a contradiction to the choice of $w$ as a word with minimal priority at $c(p')$: since $c(p')$ is even, $c(p_{j+1}) < c(p')$ and from $q_{j+1}$ there is a run that never reaches a smaller priority. Hence, $w \cdot \beta[0, j - |w|]$ would have been our choice for $w$ instead.

The second case, $c(q') < c(p')$ and $c(q')$ is odd, works almost identically so we omit the proof here. □

While this characterization of $\preceq_{\text{de}}$ seems arbitrary, it allows for an easier definition of $\equiv_{\text{de}}$ as is seen in the following statement.

**Corollary 2.2.9.** *Let $\mathcal{A}$ be a DPA with states $p, q \in Q$. Then $p \equiv_{\text{de}} q$ if and only if the following holds for all words $w \in \Sigma^*$:*

*Let $p' = \delta^*(p, w)$ and $q' = \delta^*(q, w)$. Every run that starts in $p'$ or $q'$ eventually sees a priority less than or equal to $\min\{c(p'), c(q')\}$.*

### Using delayed simulation as a merger

In general, delayed simulation does not imply equal priorities. It is not obvious if a quotient automaton can be built by choosing an arbitrary representative of each equivalence class and in fact example shows that this is not the case. However, the following theorem solves this issue.

**Lemma 2.2.10.** *Let $\mathcal{A}$ be a DPA and let $\pi$ and $\rho$ be runs of $\mathcal{A}$ on the same word $\alpha$ but starting at different states. If $\pi(0) \equiv_{\text{de}} \rho(0)$, then $\min Occ(c(\pi)) = \min Occ(c(\rho))$.*

*Proof.* Let $\pi(0) = p$ and $\rho(0) = q$. Assume towards a contradiction that $\min \text{Occ}(c(\pi)) < \min \text{Occ}(c(\rho))$. Let $k = \min \text{Occ}(c(\pi))$ and let $n$ be the first position at which $c(\pi(n)) = k$. Let $p' = \pi(n)$ and $q' = \rho(n)$. These two states are reachable from $p$ and $q$ respectively with the word $\alpha[0, n]$.

By Corollary 2.2.9, the run $\rho[n, \omega]$ must eventually see a priority at most $k$ which contradicts the assumption. □

**Theorem 2.2.11.** *Let $\mathcal{A} = (Q, \Sigma, \delta, c)$ be a DPA and let $p, q \in Q$ with $p \equiv_{\text{de}} q$ and $c(p) < c(q)$. Define $\mathcal{A}' = (Q, \Sigma, \delta, c')$ with $c'(s) = \begin{cases} c(p) & \text{if } s = q \\ c(s) & \text{else} \end{cases}$. Then $\mathcal{A} \equiv_L \mathcal{A}'$.*

*Proof.* Let $q_0 \in Q$ be an arbitrary state. We show that $L(\mathcal{A}, q_0) = L(\mathcal{A}', q_0)$.

First, consider the case that $c(p)$ is an even number. The parity of each state is at least as good in $\mathcal{A}'$ as it is in $\mathcal{A}$, so $L(\mathcal{A}, q_0) \subseteq L(\mathcal{A}', q_0)$. For the other direction, assume there is an $\alpha \in L(\mathcal{A}', q_0) \setminus L(\mathcal{A}, q_0)$, so the respective run $\rho \in Q^\omega$ is accepting in $\mathcal{A}'$ but not in $\mathcal{A}$.

For this to be true, $\rho$ must visit $q$ infinitely often and $c'(q)$ must be the lowest priority that occurs infinitely often; otherwise, the run would have the same acceptance in both automata. Thus, there is a finite word $w \in \Sigma^*$ such that from $q$, $\mathcal{A}$ reaches again $q$ via $w$ and inbetween only priorities greater than $c'(q)$ are seen.

Now consider the word $w^\omega$ and the run $\pi_q$ of $\mathcal{A}$ on said word starting in $q$. With the argument above, we know that the minimal priority occuring in $c(\pi_q)$ is greater than $c'(q)$. If we take the run $\pi_p$ on $w^\omega$ starting at $p$ though, we find that this run sees priority $c(p) = c'(q)$ at the very beginning. This contradicts Lemma 2.2.10, as $p \equiv_{\text{de}} q$. Thus, the described $\alpha$ cannot exist.

If $c(p)$ is an odd number, a very similar argumentation can be applied with the roles of $\mathcal{A}$ and $\mathcal{A}'$ reversed. We omit this repetition. □

Together with Lemma 2.1.12, this allows for a merger function that preservers language.

**Definition 2.2.4.** We define the *delayed simulation merger* as $\mu_{\text{de}} : \mathfrak{C}(\equiv_{\text{de}}) \to 2^Q$ with $\mu_{\text{de}}(\kappa) = c^{-1}(\min c(\kappa))$.

**Corollary 2.2.12.** *Let $\mathcal{A}$ be a DPA. Then $\mathcal{A}$ is language equivalent to every representative merge w.r.t. $\mu_{\text{de}}$.*

*Proof.* With Theorem 2.2.11, we can construct a language equivalent $\mathcal{A}' = (Q, \Sigma, \delta, c')$ such that for all $\kappa \in \mathfrak{C}(\equiv_{\text{de}})$, all states in $\kappa$ have the same priority in $c'$. Every representative merge of $\mathcal{A}$ w.r.t. $\mu_{\text{de}}$ is a representative merge of $\mathcal{A}'$ w.r.t. $\mu_{\text{de}}$.

In $\mathcal{A}'$, $\mu_{\text{de}}$ is the same as $\mu_{\dot{=}}^{\equiv_{\text{de}}}$. The fact that representative merges of $\mathcal{A}'$ w.r.t. $\mu_{\text{de}}$ preserve language follows from Lemma 2.1.12. □

## Resetting obligations

In the delayed simulation automaton, “obligations” correspond to good priorities that the first state has accumulated or bad priorities that the second state has accumulated and the need for the respective other state to compensate in some way. The intuitive idea behind this concept is that an obligation that cannot be compensated, stands for an infinite run in which the acceptance differs between the two states that are being compared. The issue with the original definition is that obligations carry over, even if they can only be caused finitely often. This is demonstrated in figure 2.1; the two states could be merged into one, but they are not $\equiv_{\text{de}}$-equivalent as can be seen in the delayed simulation automaton in figure 2.2.

As a solution to this, we propose a simple change to the definition of the automaton which resets the obligations every time, either state moves to a new SCC.

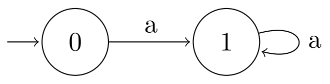

Figure 2.1: Example automaton in which the states could be merged but delayed simulation separates them.

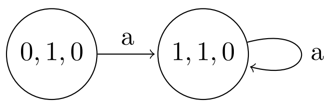

Figure 2.2: Delayed simulation automaton for 2.1.

**Definition 2.2.5.** Let $\mathcal{A} = (Q, \Sigma, q_0, \delta, c)$ be a DPA. We define the *delayed simulation automaton with SCC resets* $\mathcal{A}_{\text{deR}}(p, q) = (Q_{\text{de}}, \Sigma, \delta_{\text{deR}}, F_{\text{de}})$ with $\delta_{\text{deR}}((p, q, k), a) = \delta_{\text{de}}((p, q, \text{reset}(p, q, k, a)), a)$. Except for the addition of the reset function, this automaton is the same as $\mathcal{A}_{\text{de}}$.

If $p$ and $\delta(p, a)$ lie in the same SCC, as well as $q$ and $\delta(q, a)$, then we simply set $\text{reset}(p, q, k, a) = k$. Otherwise, i.e. if any state changes its SCC, the reset comes into play and we set $\text{reset}(p, q, k, a) = \checkmark$.

We write $p \leq_{\text{deR}} q$ if $L(\mathcal{A}_{\text{deR}}(p, q), q_0^{\text{de}}(p, q)) = \Sigma^\omega$. If also $q \leq_{\text{deR}} p$ holds, we write $p \equiv_{\text{deR}} q$.

As the definition is so similar to the original delayed simulation, most results that we have already proven translate directly to the new relation.

**Theorem 2.2.13.** *$\equiv_{deR}$ is a congruence relation.*

**Lemma 2.2.14.** *$\leq_{de} \subseteq \leq_{deR}$.*

*Proof.* Consider the two simulation automata $\mathcal{A}_{\text{de}}(p, q)$ and $\mathcal{A}_{\text{deR}}(p, q)$ and let $\alpha \in \Sigma^\omega$ be an arbitrary word. Let $(p_i, q_i, k_i)_{i \in \mathbb{N}}$ and $(p_i, q_i, l_i)_{i \in \mathbb{N}}$ be the runs of these two automata on $\alpha$. We claim that $k_i \leq_\checkmark l_i$ at every position. Then, since $L(\mathcal{A}_{\text{de}}(p, q), q_0^{\text{de}}(p, q)) = \Sigma^\omega$, both runs must be accepting.

We know $k_0 = l_0$ by definition. For the sake of induction, we look at position $i + 1$. If neither $p_i$ nor $q_i$ change their SCC in this step, the statement follows from lemma 2.2.1. Otherwise, $l_{i+1} = \checkmark \geq_\checkmark k_{i+1}$. □

The real difference comes up when looking at theorem 2.2.11. If we consider again the example given in figure 2.1, if that theorem would hold the same for $\equiv_{\text{deR}}$ we could assign both states the priority 0, which would then make the automaton accept every word. This is obviously not the same as the original automaton, so this statement deserves some additional inspection.

**Lemma 2.2.15.** *Let $\mathcal{A}$ be a DPA and let $\pi$ and $\rho$ be runs of $\mathcal{A}$ on the same word but starting at different states. Let $\mathit{off}_\pi$ and $\mathit{off}_\rho$ be the positions after which $\pi$ and $\rho$ respectively only stay in one single SCC. Let $\mathit{off} = \max\{\mathit{off}_\pi, \mathit{off}_\rho\}$. If $\pi(0) \equiv_{deR} \rho(0)$, then $\min Occ(c(\pi[\mathit{off}, \omega])) = \min Occ(c(\rho[\mathit{off}, \omega]))$.*

*Proof.* Let $k = \min \text{Occ}(c(\pi[\text{off}, \omega]))$ and $l = \min \text{Occ}(c(\rho[\text{off}, \omega]))$. Assume towards a contradiction without loss of generality that $k < l$. Let $\alpha$ be the word that is read by the two runs.

If $k$ is even, let $\sigma$ be the run of $\mathcal{A}_{\text{de}}(\pi(0), \rho(0))$ on $\alpha$. Let $n \geq \text{off}$ be a position at which $c(\pi(n)) = k$. We claim that for all $i \geq n$, the third component of $\sigma(i)$ is $k$.

At $\sigma(n)$, this must be true because $k < l \leq c(\rho(n))$ and thus $c(\rho(n)) \not\preceq_p c(\pi(n))$. At all positions after $n$, it can never occur that $c(\rho(i)) \leq k$ or that $c(\pi(i))$ is odd and smaller than $k$. There is also never a change in SCCs anymore, by choice of $n$. The rest follows from the definition of $\gamma$.

If $k$ is odd, we can argue similarly on the run of $\mathcal{A}_{\mathrm{de}}(\rho(0), \pi(0))$. As soon as $c(\pi)$ reaches its minimum, the third component of the run will never change again. $\square$

**Theorem 2.2.16.** *Let $\mathcal{A} = (Q, \Sigma, q_0, \delta, c)$ be a DPA and let $p, q \in Q$ with $p \equiv_{\mathrm{de}} q$ and $c(p) < c(q)$. Define $\mathcal{A}' = (Q, \Sigma, q_0, \delta, c')$ with $c'(s) = \begin{cases} c(p) & \text{if } s = q \\ c(s) & \text{else} \end{cases}$. If $\{p\}$ is not a trivial SCC in $\mathcal{A}$, then $\mathcal{A} \equiv_L \mathcal{A}'$.*

*Proof.* Let $q_0 \in Q$ be an arbitrary state. We show that $L(\mathcal{A}, q_0) = L(\mathcal{A}', q_0)$.

First, consider the case that $c(p)$ is an even number. The parity of each state is at least as good in $\mathcal{A}'$ as it is in $\mathcal{A}$, so $L(\mathcal{A}, q_0) \subseteq L(\mathcal{A}', q_0)$. For the other direction, assume there is an $\alpha \in L(\mathcal{A}', q_0) \setminus L(\mathcal{A}, q_0)$, so the respective run $\rho \in Q^\omega$ is accepting in $\mathcal{A}'$ but not in $\mathcal{A}$.

For this to be true, $\rho$ must visit $q$ infinitely often and $c'(q)$ must be the lowest priority that occurs infinitely often; otherwise, the run would have the same acceptance in both automata. Thus, there is a finite word $w \in \Sigma^*$ such that from $q$, $\mathcal{A}$ reaches again $q$ via $w$ and inbetween only priorities greater than $c'(q)$ are seen.

Now consider the word $w^\omega$ and the run $\pi_q$ of $\mathcal{A}$ on said word starting in $q$. With the argument above, we know that the minimal priority occuring in $c(\pi_q)$ is greater than $c'(q)$. If we take the run $\pi_p$ on $w^\omega$ starting at $p$ though, we find that this run sees priority $c(p) = c'(q)$ at the very beginning. This contradicts Lemma 2.2.10, as $p \equiv_{\mathrm{de}} q$. Thus, the described $\alpha$ cannot exist.

If $c(p)$ is an odd number, a very similar argumentation can be applied with the roles of $\mathcal{A}$ and $\mathcal{A}'$ reversed. We omit this repetition. $\square$

**Lemma 2.2.17.** *Let $\mathcal{A} = (Q, \Sigma, \delta, c)$ be a DPA and let $\{s\} \subseteq Q$ be a trivial SCC in $\mathcal{A}$. For any arbitrary $k \in \mathbb{N}$, let $\mathcal{A}' = (Q, \Sigma, \delta, c')$ be another DPA with $c' \restriction_{Q \setminus \{s\}} = c \restriction_{Q \setminus \{s\}}$ and $c'(s) = k$. Then $\mathcal{A} \equiv_L \mathcal{A}'$.*

*Proof.* Let $q \in Q$ be any state. We show that $L(\mathcal{A}, q) = L(\mathcal{A}', q)$. Let $\rho$ and $\rho'$ be the runs of $\mathcal{A}$ and $\mathcal{A}'$ starting in $q$ on some $\alpha \in \Sigma^\omega$. As $s$ lies in a trivial SCC, the runs visit that state at most once. Therefore, $\mathrm{Inf}c(\rho) = \mathrm{Inf}c'(\rho')$ and the runs have the same acceptance. $\square$

Using these results, we can now define a suitable merger function for delayed simulation with resets.

**Definition 2.2.6.** We define the *delayed simulation merger with resets* $\mu_{\mathrm{deR}} : \mathfrak{C}(\equiv_{\mathrm{deR}}) \to 2^Q$ as follows: Let $T \subseteq Q$ be the set of states that are in trivial SCCs. For every equivalence class $\kappa$ of $\equiv_{\mathrm{deR}}$, we set $\mu_{\mathrm{deR}}(\kappa) = c(\min c^{-1}(\kappa \setminus T))$.

**Corollary 2.2.18.** *Let $\mathcal{A}$ be a DPA. Then $\mathcal{A}$ is language equivalent to every representative merge w.r.t. $\mu_{\mathrm{de}}$.*

*Proof.* With Theorem 2.2.11, we can construct a language equivalent $\mathcal{A}' = (Q, \Sigma, \delta, c')$ such that for all $\kappa \in \mathfrak{C}(\equiv_{\mathrm{de}})$, all states in $\kappa$ have the same priority in $c'$. Every representative merge of $\mathcal{A}$ w.r.t. $\mu_{\mathrm{de}}$ is a representative merge of $\mathcal{A}'$ w.r.t. $\mu_{\mathrm{de}}$.

In $\mathcal{A}'$, $\mu_{\mathrm{de}}$ is the same as $\mu_{\div}^{\equiv_{\mathrm{de}}}$. The fact that representative merges of $\mathcal{A}'$ w.r.t. $\mu_{\mathrm{de}}$ preserve language follows from Lemma 2.1.12. $\square$

Finally, we provide a quick analysis of the computational effort for delayed simulation.

**Theorem 2.2.19.** *For a given DPA $\mathcal{A}$, $\equiv_{\mathrm{de}}$ and $\equiv_{\mathrm{deR}}$ can be computed in $\mathcal{O}(|Q|^2 \cdot |c(Q)|)$.*

*Proof.* $\mathcal{A}_{\mathrm{de}}$ and $\mathcal{A}_{\mathrm{deR}}$ are of size $\mathcal{O}(|Q|^2 \cdot |c(Q)|)$. Building $\mathcal{A}_{\mathrm{de}}$ is straight-forward and $\mathcal{A}_{\mathrm{deR}}$ also only requires additional information about the SCCs of $\mathcal{A}$, which can be computed in linear time.

Once the delayed simulation automata are built, we have to solve the universal problem, i.e. find all states from which all words are accepted. For DBAs this can be done in linear time with depth search or similar algorithms. $\square$

# Bibliography

[1] Carsten Fritz and Thomas Wilke. Simulation relations for alternating büchi automata. *Theor. Comput. Sci.*, 338(1-3):275–314, June 2005.

[2] Monika Rauch Henzinger and Jan Arne Telle. Faster algorithms for the nonemptiness of streett automata and for communication protocol pruning. In *Algorithm Theory — SWAT'96*, pages 16–27, Berlin, Heidelberg, 1996. Springer Berlin Heidelberg.

[3] 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.