Chapter 1

Basic Definitions

The first chapter (re)defines the fundamentals of this thesis and notation used later on.

1.1 General Mathematical Terms

We begin with a set theoretic approach to define the natural numbers.

Definition 1.1.1. The natural numbers $\mathbb{N} = \{0, 1, 2, 3, \dots\}$ are defined as follows: For every natural n, we set $n := \{0, \dots, n-1\}$. For example, $0 = \emptyset$, $1 = \{0\}$, $2 = \{0, 1\} = \{\emptyset, \{\emptyset\}\}$.

We can extend this definition to ordinal numbers up to ω.

Definition 1.1.2. We write $\omega = \mathbb{N}$. The set of *ordinal numbers* (up to ω) is $\omega \cup \{\omega\} = \{0, 1, 2, \dots, \omega\}$.
For two ordinal numbers α, β, we define $\alpha < \beta$ if and only if $\alpha \in \beta$.
For an ordinal $\alpha \in \omega$, we define $\alpha^+ = \alpha \cup \{\alpha\}$ as the successor of α. If $\alpha = \beta^+$ for some ordinal β, we define $\alpha^- = \beta$. Otherwise, i.e. if $\alpha \in \{0, \omega\}$, we define $\alpha^- = \alpha$.

Definition 1.1.3. Let α, β be ordinal numbers. We need to consider the following special cases of ordinal arithmetic:

- If $\alpha, \beta \in \mathbb{N}$, then $\alpha + \beta$ is defined as usual.
- If $\alpha = \omega, \beta = 0$, then $\alpha + \beta = \omega$.
- If $\alpha \in \mathbb{N}, \beta = \omega$, then $\alpha + \beta = \omega$.

In particular $1 + \alpha = \begin{cases} \alpha^+ & \text{if } \alpha < \omega \\ \omega & \text{if } \alpha = \omega \end{cases}$.

For some $n \in \mathbb{N}$, we write $\alpha - n = \alpha$ if $n = 0$, and $\alpha - n = (\alpha^-) - (n-1)$ otherwise.

Definition 1.1.4. Let X and Y be two sets. We define X^Y as the set of all functions $f : Y \to X$.

Definition 1.1.5. Let X be a set. We denote the power set $\{Y \mid Y \subseteq X\}$ by $\mathcal{P}(X)$.

Definition 1.1.6. A *partial order* on a set X is a relation $< \subseteq X \times X$ which satisfies

- irreflexivity, i.e. $x \not< x$ for all $x \in X$, and
- transitivity, i.e. if $x < y$ and $y < z$, then $x < z$.

$<$ is called a *linear order* if additionally to that it satisfies totality, i.e. for all $x, y \in X$, either $x = y$, or $x < y$, or $y < x$.

$<$ is called a *well order* if there is no infinitely descending sequence in X.

Definition 1.1.7. Let $f : X \to Y$ be a function and $X' \subseteq X$. We define $f(X') = \{y \in Y \mid$ There is an $x \in X'$ such that $f(x) = y.\}$, and $f \restriction_{X'} = g$, where $g : X' \to Y$ is a function with $g(x) = f(x)$.

Definition 1.1.8. Let X, Y be sets. We define $X \dot\cup Y = Z$ as the *disjunct union* of X and Y, which means we assume via renaming that $X \cap Y = \emptyset$.

1.2 Words and Languages

Definition 1.2.1. A non-empty set of symbols can be called *alphabet* and will usually be denoted by a variable Σ, Γ, Π.
A finite *word* (denoted by w, v, u) over an alphabet Σ is a function $w : \{0, \dots, n-1\} \to \Sigma$ for some $n \in \mathbb{N}$. The *length* of this word is $|w| = n$. The word w with length $|w| = 0$ is called the *empty word* ε.
This gives an alternative description of Σ^n: $\Sigma^n = \{w : n \to \Sigma \mid w$ is a word of length n over $\Sigma\}$.
Further, we define $\Sigma^* = \bigcup_{n \in \mathbb{N}} \Sigma^n$ as the set of all words over Σ.
A *language* (denoted by L, K) over an alphabet Σ is a set $L \subseteq \Sigma^*$.

We now repeat the extension of this definition for infinite words.

Definition 1.2.2. An *ω-word* (denoted by α, β, γ) over an alphabet Σ is a function $\alpha : \mathbb{N} \to \Sigma$. The length of α is $|\alpha| = \omega = \mathbb{N}$.
Again, this can be used to describe Σ^ω as $\{w : \mathbb{N} \to \Sigma \mid w$ is an ω-word over $\Sigma\}$.
An *ω-language* (denoted by U, V) over an alphabet Σ is a set $L \subseteq \Sigma^\omega$.

We can now define the usual operations on words and languages. From now on, we always assume Σ to be an arbitrary alphabet unless specified otherwise.

Definition 1.2.3. Let $v, w \in \Sigma^*$ and $w_i \in \Sigma^*$ for all $i \in \mathbb{N}$ be words over Σ and $\alpha \in \Sigma^\omega$ be an ω-word over Σ.

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 α (denoted by $w \cdot \alpha$) is an ω-word β 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 *ω-iteration* of w (denoted by w^ω) is an ω-word α such that:

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

The *concatenation* of a sequence $(w_i)_{i \in \mathbb{N}}$ (denoted by $\bigcirc_{i \in \mathbb{N}} w_i$ or $w_0 \cdot \dots$) is an ω-word α such that:

$$\alpha : \mathbb{N} \to \Sigma, i \mapsto w_j(x)$$

where $x = i - \sum_{k=0}^{j-1} |w_k|$ and j is the biggest number such that $x \geq 0$.

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 1.2.4. Let $L, K \subseteq \Sigma^*$ be a language and $U \subseteq \Sigma^\omega$ be an ω-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 $L^* = \bigcup_{n \in \mathbb{N}} L^n$.
The *ω-iteration* of L is $L^\omega = \{\alpha \in \Sigma^\omega \mid \text{There is a sequence } (w_i)_{i \in \mathbb{N}} \in L \text{ such that } \alpha = \bigcirc_{i \in \mathbb{N}} w_i\}$.
The *∞-iteration* of L is $L^\infty = L^* \cup L^\omega$.

Definition 1.2.5. Let $w \in \Sigma^\infty$ be a word. For $0 \leq n, m < |w|$, we define $w[n, m] = w(n) \cdots w(m)$.
If $w \in \Sigma^*$ is finite, we define $\text{Tail}(w) = w[1, |w| - 1]$ and $\text{Last}(w) = w(|w| - 1)$.
If $w \in \Sigma^\omega$ is infinite, we also define $w[n, \omega] = \bigcirc_{i \geq n} w(i)$ and $\text{Tail}(w) = w[1, \omega]$.

Definition 1.2.6. The *occurrence set* of a word $w \in \Sigma^\infty$ is the set of symbols which occur at least once in w.

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

Definition 1.2.7. The *infinity set* of a word $w \in \Sigma^\infty$ is the set of symbols which occur infinitely often in w.

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

Note that for finite alphabets, $\text{Inf}(w) = \emptyset$ iff $w \in \Sigma^*$.

1.3 Syntax of Regular Expressions

We now define three kinds of regular expressions for the languages. For this entire section, we assume Σ to be an arbitrary alphabet.

Definition 1.3.1. A *regular expression* is defined inductively as follows.

- ε and $\emptyset$ are regular expressions.
- For every $a \in \Sigma$, a is a regular expression.
- If r and s are regular expressions, so are $(r \cdot s)$ and $(r + s)$.
- If r is a regular expression, so is (r^*).

For readability, unnecessary brackets will be omitted from this notation.
We call $\mathcal{R}eg_*[\Sigma]$ the set of all regular expressions over Σ, whereas the specification of the alphabet will be left out if it is clear within the given context.

Definition 1.3.2. Let $r \in \mathcal{R}eg_*$ be a regular expression. The set of *sub-expressions* $\mathrm{sub}(r)$ is the set of all regular expressions "contained" in r and is defined inductively as:

- If $r \in \{\varepsilon, \emptyset\}$, then $\mathrm{sub}(r) = \{r\}$.
- If $r = a$ for some $a \in \Sigma$, then $\mathrm{sub}(r) = \{a\}$.
- If $r = s \cdot t$, then $\mathrm{sub}(r) = \{r\} \cup \mathrm{sub}(s) \cup \mathrm{sub}(t)$.
- If $r = s + t$, then $\mathrm{sub}(r) = \{r\} \cup \mathrm{sub}(s) \cup \mathrm{sub}(t)$.
- If $r = s^*$, then $\mathrm{sub}(r) = \{r\} \cup \mathrm{sub}(s)$.

Example 1.1. For $\Sigma = \{a, b, c\}$ and the regular expression $r = ((ab) + (b + c^*))^*$, the set of sub-expressions is $\mathrm{sub}(r) = \{r, (ab) + (b + c^*), ab, b + c^*, c^*, a, b, c\}$.

Definition 1.3.3. An *ω-regular expression* is an expression $r = r_1 s_1^\omega + \cdots + r_n s_n^\omega$ for regular expressions $r_1, s_1, \ldots, r_n, s_n \in \mathcal{R}eg_*$.
We call the set of all ω-regular expressions $\mathcal{R}eg_\omega[\Sigma]$.

Definition 1.3.4. Let $r = r_1 s_1^\omega + \cdots + r_n s_n^\omega$ be an ω-regular expression. The set of sub-expressions of r are defined similarly as before:

- If $n = 1$, then $\mathrm{sub}(r) = \{r, s_1^\omega\} \cup \mathrm{sub}(r_1) \cup \mathrm{sub}(s_1)$.
- If $n > 1$, then $\mathrm{sub}(r) = \{r\} \cup \mathrm{sub}(r_1 s_1^\omega + \cdots + r_{n-1} s_{n-1}^\omega) \cup \mathrm{sub}(r_n s_n^\omega)$.

Example 1.2. For $\Sigma = \{a, b, c\}$ and the regular expression $r = ab^\omega + \varepsilon(cc)^\omega$, the set of sub-expressions is $\mathrm{sub}(r) = \{r, ab^\omega, \varepsilon(cc)^\omega, b^\omega, (cc)^\omega, cc, a, b, c, \epsilon\}$.

Definition 1.3.5. An *∞-regular expression* is an extension of regular expression, i.e. it is defined inductively with the same rules as regular expressions with the addition of a new one:

- If r is an ∞-regular expression, so is (r^∞).

The set of all ∞-regular expressions is $\mathcal{R}eg_\infty[\Sigma]$.

Definition 1.3.6. We set $\mathcal{R}eg[\Sigma] = \mathcal{R}eg_*[\Sigma] \cup \mathcal{R}eg_\omega[\Sigma] \cup \mathcal{R}eg_\infty[\Sigma]$.

Definition 1.3.7. The sub-expressions for ∞-regular expressions are defined as they are for regular expressions, with the addition that $\mathrm{sub}(r^\infty) = \{r^\infty\} \cup \mathrm{sub}(r)$.
A sub-expression $s \in \mathrm{sub}(r)$ of an ∞-regular expression r is called an *∞-base* of r if $s^\infty \in \mathrm{sub}(r)$. s is called a *$*$-base* of r if $s^* \in \mathrm{sub}(r)$.
We call s a *$*$-sub-expression* if $s = r^*$ for some $*$-base r. Analogous to that we call s an *∞-sub-expression* if $s = r^\infty$ for some ∞-base r.

Example 1.3. For $\Sigma = \{a, b, c\}$ and the regular expression $r = (ab^\infty a)^\infty + c^*$, the ∞-bases of r are b, $ab^\infty a$. The only $*$-base of r is c.

Definition 1.3.8. For two regular expressions of an arbitrary type r, s we set $r \in_{\mathrm{reg}} s$ if $r \neq s$ and $r \in \mathrm{sub}(s)$.

Definition 1.3.9. Let $r \in \mathcal{R}_e g[\Sigma]$ be a regular expression. We define the *length* of r ($|r|$) as the number of Σ-symbols.

- $|\varepsilon| = |\emptyset| = 0$
- $|a| = 1$ for all $a \in \Sigma$
- $|r + s| = |r \cdot s| = |r| + |s|$
- $|r^*| = |r^\omega| = |r^\infty| = |r|$

1.4 Semantics of Regular Expressions

We now define the languages of the three kinds of regular expressions we introduced.

Definition 1.4.1. Let $r \in \mathcal{R}eg_*[\Sigma]$ be a regular expression. The *language of* r, $L(r) \subseteq \Sigma^*$, is defined inductively as:

- $L(\emptyset) = \emptyset$
- $L(\varepsilon) = \{\varepsilon\}$
- For all $a \in \Sigma$, $L(a) = \{a\}$
- $L(r + s) = L(r) \cup L(s)$
- $L(r \cdot s) = L(r) \cdot L(s)$
- $L(r^*) = (L(r))^*$

A language $L \subseteq \Sigma^*$ is called *regular* if there is a regular expression $r \in \mathcal{R}eg_*$ such that $L = L(r)$.

Definition 1.4.2. Let $r = r_1 s_1^\omega + \cdots + r_n s_n^\omega$ be an ω-regular expression. $L(r) \subseteq \Sigma^\omega$ is defined as $L(r) = \bigcup_{i=1}^{n} L(r_i) \cdot (L(s_i))^\omega$.
An ω-language $U \subseteq \Sigma^\omega$ is called *ω-regular* if there is an ω-regular expression $r \in \mathcal{R}eg_\omega$ such that $U = L(r)$.

Definition 1.4.3. Let $r \in \mathcal{R}eg_\infty[\Sigma]$ be an ∞-regular expression. The language of r, $L(r) \subseteq \Sigma^\infty$, is defined inductively as:

- $L(\emptyset) = \emptyset$
- $L(\varepsilon) = \{\varepsilon\}$
- For all $a \in \Sigma$, $L(a) = \{a\}$
- $L(r + s) = L(r) \cup L(s)$
- $L(r \cdot s) = (L_*(r) \cdot L(s)) \cup L_\omega(r)$
- $L(r^*) = ((L_*(r))^*) \cdot (\{\varepsilon\} \cup L_\omega(r))$
- $L(r^\infty) = L(r^*) \cup (L_*(r))^\omega$

Here, $L_*(r)$ and $L_\omega(r)$ denote the sets of finite and infinite words in $L(r)$, i.e. $L_*(r) = L(r) \cap \Sigma^*$ and $L_\omega(r) = L(r) \cap \Sigma^\omega$.

A language $L \subseteq \Sigma^\infty$ is called *∞-regular* if there is an ∞-regular expression $r \in \mathcal{R}eg_\infty$ such that $L = L(r)$.

Example 1.4. Let $\Sigma = \{a, b\}$ and $r = a^\infty b^\infty$. Then $L_*(r) = L(a^*b^*)$ and $L_\omega(r) = L(a^\omega + a^*b^\omega)$.

Definition 1.4.4. We say for any two types of regular expressions $r, s \in \mathcal{R}eg[\Sigma]$, r and s are *equivalent* ($r \equiv s$) if $L(r) = L(s)$.

1.4.1 Correlation of regular, ω-regular, and ∞-regular expressions

While regular expressions and ω-regular expressions are the standard object used in their respective fields of automata theory, ∞-regular expressions are just as powerful as we show in the following statements.

Lemma 1.4.1. *Let $r \in \mathcal{R}eg_\infty$ be an ∞-regular expression. There is a regular expression $g(r) \in \mathcal{R}eg_*$ such that $L_*(r) = L(g(r))$ and $|g(r)| \in \mathcal{O}(|r|)$.*

Proof. We define the construction of $g(r)$ inductively.

- If $r = a \in \Sigma \cup \{\varepsilon\}$, then $g(r) = r$.
  $L_*(r) = \{a\} = L(g(r))$.
- If $r = \emptyset$, then $g(r) = r$.
  $L_*(r) = \emptyset = L(g(r))$.

- If $r = s + t$, then $g(r) = g(s) + g(t)$.

$$\begin{aligned} & L_*(r) \\ &= (L(s) \cup L(t)) \cap \Sigma^* \\ &= (L(s) \cap \Sigma^*) \cup (L(t) \cap \Sigma^*) \\ &= L_*(s) \cup L_*(t) \\ &= L(g(r)) \end{aligned}$$

- If $r = s \cdot t$, then $g(r) = g(s) \cdot g(t)$.

$$\begin{aligned} & L_*(r) \\ &= ((L_*(s) \cdot L(t)) \cup L_\omega(s)) \cap \Sigma^* \\ &= ((L_*(s) \cdot L(t)) \cap \Sigma^*) \cup (L_\omega(s)) \cap \Sigma^*) \\ &= (L_*(s) \cdot L(t)) \cap \Sigma^* \\ &= (L_*(s) \cap \Sigma^*) \cdot (L(t) \cap \Sigma^*) \\ &= L_*(s) \cdot L_*(t) \\ &= L(g(r)) \end{aligned}$$

- If $r = s^*$, then $g(r) = g(s)^*$.

$$\begin{aligned} & L_*(r) \\ &= (((L_*(s))^*) \cdot (\{\varepsilon\} \cup L_\omega(s))) \cap \Sigma^* \\ &= (((L_*(s))^* \cap \Sigma^*) \cdot ((\{\varepsilon\} \cap \Sigma^*) \cup (L_\omega(s) \cap \Sigma^*)) \\ &= (((L_*(s))^*) \cdot (\{\varepsilon\} \cup \emptyset) \\ &= ((L_*(s))^* \\ &= L(g(r)) \end{aligned}$$

- If $r = s^\infty$, then $g(r) = g(s)^*$.

$$\begin{aligned} & L_*(r) \\ &= (L(s^*) \cup (L_*(s))^\omega) \cap \Sigma^* \\ &= (L(s^*) \cap \Sigma^*) \cup (L_*(s))^\omega \cap \Sigma^*) \\ &= L_*(s^*) \cup \emptyset \\ &= L(g(s^*)) \\ &= L(g(r)) \end{aligned}$$

□

Lemma 1.4.2. *Let $r \in \mathcal{R}eg_\infty$ be an ∞-regular expression. There is an ω-regular expression $h(r) \in \mathcal{R}eg_\omega$ such that $L_\omega(r) = L(h(r))$.*

Proof. We define the construction of $h(r)$ inductively.

- If $r = a \in \Sigma \cup \{\varepsilon, \emptyset\}$, then $h(r) = \emptyset$.
  $L_\omega(r) = \emptyset = L(h(r))$
- If $r = s + t$, then $h(r) = h(s) + h(t)$.

$$\begin{aligned} & L_\omega(r) \\ &= (L(s) \cup L(t)) \cap \Sigma^\omega \\ &= (L(s) \cap \Sigma^\omega) \cup (L(t) \cap \Sigma^\omega) \\ &= L_\omega(s) \cup L_\omega(t) \\ &= L(h(s)) \cup L(h(t)) \\ &= L(h(r)) \end{aligned}$$

- If $r = s \cdot t$, let $h(t) = a_1 b_1^\omega + \cdots + a_n b_n^\omega$ and set $h(r) = h(s) + g(s)a_1 b_1^\omega + \cdots + g(s)a_n b_n^\omega$.

$$\begin{aligned} & L_\omega(r) \\ &= ((L_*(s) \cdot L(t)) \cup L_\omega(s)) \cap \Sigma^\omega \\ &= ((L_*(s) \cdot L(t)) \cap \Sigma^\omega) \cup (L_\omega(s) \cap \Sigma^\omega) \\ &= ((L_*(s) \cdot L(t)) \cap \Sigma^\omega) \cup L_\omega(s) \\ &= ((L_*(s) \cap \Sigma^\omega) \cdot (L(t) \cap \{\varepsilon\})) \cup ((L_*(s) \cap \Sigma^*) \cdot (L(t) \cap \Sigma^\omega)) \cup L_\omega(s) \\ &= (L_*(s) \cdot (L(t) \cap \Sigma^\omega)) \cup L_\omega(s) \\ &= (L_*(s) \cdot L_\omega(t)) \cup L_\omega(s) \\ &= (L(g(s)) \cdot L(h(t))) \cup L(h(s)) \\ &= (L(g(s)) \cdot L(a_1 b_1^\omega + \cdots + a_n b_n^\omega)) \cup L(h(s)) \\ &= (L(g(s)a_1 b_1^\omega + \cdots + g(s)a_n b_n^\omega)) \cup L(h(s)) \\ &= L(h(r)) \end{aligned}$$

- If $r = s^*$, then $h(r) = g(s^*) \cdot h(s)$.

$$\begin{aligned} & L_\omega(r) \\ &= (((L_*(s))^*) \cdot (\{\varepsilon\} \cup L_\omega(s))) \cap \Sigma^\omega \\ &= (((L_*(s))^*) \cap \Sigma^\omega) \cup ((((L_*(s))^*) \cap \Sigma^*) \cdot (L_\omega(s) \cap \Sigma^\omega)) \\ &= (((L_*(s))^*) \cap \Sigma^*) \cdot (L_\omega(s) \cap \Sigma^\omega) \\ &= ((L_*(s))^*) \cdot L_\omega(s) \\ &= L_*(s^*) \cdot L_\omega(s) \\ &= L(g(s^*)) \cdot L(h(s)) \\ &= L(h(r)) \end{aligned}$$

- If $r = s^\infty$, then $h(r) = h(r^*) + g(s)^\omega$.

$$\begin{aligned} & L_\omega(r) \\ &= (L(s^*) \cup (L_*(s))^\omega) \cap \Sigma^\omega \\ &= (L(s^*) \cap \Sigma^\omega) \cup ((L_*(s))^\omega \cap \Sigma^\omega) \\ &= L_\omega(s^*) \cup (L_*(s))^\omega \\ &= L(h(s^*)) \cup L(g(s))^\omega \\ &= L(h(r)) \end{aligned}$$

□

Proposition 1.4.3. *A language $L \subseteq \Sigma^*$ is regular if and only if there is an ∞-regular expression $r \in \mathcal{R}eg_\infty$ such that $L_*(r) = L$.*

Proof. **Only if**
If L is regular, there is a regular expression $r \in \mathcal{R}eg_*$ with $L = L(r)$. By construction, r is also an ∞-regular expression with $L(r) = L$.

If
Let $r \in \mathcal{R}eg_\infty$ be the described ∞-regular expression. With Lemma **??**, $L = L(g(r))$ is regular.

□

Proposition 1.4.4. *An ω-language $U \subseteq \Sigma^\omega$ is ω-regular if and only if there is an ∞-regular expression $r \in \mathcal{R}eg_\infty$ such that $L_\omega(r) = U$.*

Proof. **Only if**
If L is ω-regular, there is a regular expression $r \in \mathcal{R}eg_\omega$ with $L = L(r)$. By replacing every occurrence of an s^ω by s^∞, the ∞-regular expression r' is constructed. By definition, $L(r') = L(r) = L$.

If
Let $r \in \mathcal{R}eg_\infty$ be the described ∞-regular expression. With Lemma **??**, $L = L(h(r))$ is ω-regular.

□

Proposition 1.4.5. *Let $L \subseteq \Sigma^\infty$ be a language. L is ∞-regular if and only if $L \cap \Sigma^*$ is regular and $L \cap \Sigma^\omega$ is ω-regular.*

Proof. **Only if** Let L be ∞-regular, so there is an ∞-regular expression $r \in \mathcal{R}eg_\infty[\Sigma]$ such that $L(r) = L$. Using Lemmas **??** and **??**, $L \cap \Sigma^* = L(g(r))$ and $L \cap \Sigma^\omega = L(h(r))$.

If Let $L \cap \Sigma^*$ be regular and $L \cap \Sigma^\omega$ be ω-regular, so there are a regular expressions $s \in \mathcal{R}eg_*[\Sigma]$ and an ω-regular expression $t \in \mathcal{R}eg_\omega[\Sigma]$ such that $L(s) = L \cap \Sigma^*$ and $L(t) = L \cap \Sigma^\omega$. Our goal is to construct an ∞-expression r with $L(r) = L(s) \cup L(t) = L$.

We construct t' from t by replacing all $^\omega$ with $^\infty$ and set $r = s + t' \cdot \emptyset$. It should be clear that $L(t) = L_\omega(t')$.

$$\begin{aligned} & L(r) \\ = \; & L(s) \cup L(t' \cdot \emptyset) \\ = \; & L(s) \cup (L_*(t') \cdot \emptyset) \cup L_\omega(t') \\ = \; & L(s) \cup L_\omega(t') \\ = \; & L(s) \cup L(t) \\ = \; & L \end{aligned}$$

□

1.5 Automata

In this work, we focus ourselves on the model of parity automata (abbreviated PA) (also known as Rabin chain automata). We now give three different but similar definitions of this structure, the classical model and two variations better suited for the upcoming constructions.

Definition 1.5.1. A *standard parity automaton* is a tuple $\mathcal{A} = (Q, \Sigma, q_0, \delta, c)$, where

- Q is the finite, nonempty set of states.
- Σ is the input alphabet.
- $q_0 \in Q$ is the initial state.
- $\delta : Q \times \Sigma \to \mathcal{P}(Q)$ is the transition function.
- $c : Q \to \mathbb{N}$ is the priority function.

$\mathcal{A}$ is said to be *deterministic* if for all $q \in Q, a \in \Sigma$, the transition is uniquely defined, i.e. $|\delta(q, a)| \leq 1$. In that case, we write $\delta(q, a) = \emptyset$ or $\delta(q, a) = p \in Q$.

A *run* of $\mathcal{A}$ on an ω-word $\alpha \in \Sigma^\omega$ is a word $\rho \in Q^\omega$, such that $\rho(0) = q_0$ and for all $i > 0$: $\rho(i) \in \delta(\rho(i-1), \alpha(i))$.
The priorities of ρ are $c(\rho) = \gamma \in \mathbb{N}^\omega$ such that $\gamma(i) = c(\rho(i))$. ρ is *accepting* if $\max \mathrm{Inf}(c(\rho))$ is an even number.

The language of $\mathcal{A}$, written $L(\mathcal{A}) \subseteq \Sigma^\omega$ is defined as the set of all ω-words over Σ which have an accepting run on $\mathcal{A}$.
$\mathcal{A}$ *recognizes* or *accepts* a language $U \subseteq \Sigma^\omega$ if $L(\mathcal{A}) = U$.

Definition 1.5.2. A *transition parity automaton* is a tuple $\mathcal{A} = (Q, \Sigma, q_0, \delta, c, F)$, where

- Q is the finite, nonempty set of states.
- Σ is the input alphabet.
- $q_0 \in Q$ is the initial state.
- $\delta : Q \times \Sigma \to \mathcal{P}(Q)$ is the transition function.

- $c : Q \times Q \to \mathbb{Z}$ is the priority function.
- $F \subseteq Q$ is the set of accepting states.

Determinism of $\mathcal{A}$ is defined as for standard parity automata.

A *run* of $\mathcal{A}$ on an ∞-word $w \in \Sigma^\infty$ is a word $\rho \in Q^{1+|w|}$, such that $\rho(0) = q_0$ and for all $0 < i \leq |w| : \rho(i) \in \delta(\rho(i-1), \alpha(i))$.
The priorities of ρ are $c(\rho) = \gamma \in \mathbb{N}^{|w|}$ such that $\gamma(i) = c(\rho(i), \rho(i+1))$. For an ω-word, ρ is *accepting* if $\max \mathrm{Inf}(c(\rho))$ is an even number. For a finite word, ρ is *accepting* if $\rho(|w|) \in F$.

The language of $\mathcal{A}$, written $L(\mathcal{A}) \subseteq \Sigma^\infty$ is defined as the set of all ∞-words over Σ which have an accepting run on $\mathcal{A}$. We define $L_*(\mathcal{A}) = L(\mathcal{A}) \cap \Sigma^*$ and $L_\omega(\mathcal{A}) = L(\mathcal{A}) \cap \Sigma^\omega$.
$\mathcal{A}$ *recognizes* or *accepts* a language $L \subseteq \Sigma^\infty$ if $L(\mathcal{A}) = L$.

Regarding the transition function δ, we also write $\delta(q, A) = \bigcup_{a \in A} \delta(q, a)$ for some set of symbols $A \subseteq \Sigma$.

Definition 1.5.3. An *expression parity automaton* is a tuple $\mathcal{A} = (Q, \Sigma, q_0, \delta, c)$, where

- Q is the finite, nonempty set of states.
- Σ is the input alphabet.
- $q_0 \in Q$ is the initial state.
- $\delta : Q \times Q \to \mathcal{R}eg_\infty[\Sigma]$ is the transition function, with ∞-regular expressions as labels.
- $c : Q \to \mathbb{N}$ is the priority function.

A *run* of $\mathcal{A}$ on an ∞-word $w \in \Sigma^\infty$ is a word $\rho \in Q^{1+|w|}$, such that $\rho(0) = q_0$ and there is a decomposition $w = \bigcirc_{i<|\rho|} v_i$, with $v_i \in \begin{cases} \Sigma^* & \text{if } i+1 < |\rho| \\ \Sigma^\infty & \text{else} \end{cases}$, such that $v_i \in L(\delta(\rho(i), \rho(i+1)))$ for all i.
The priorities of ρ are $c(\rho) = \gamma \in \mathbb{N}^{|\rho|}$ such that $\gamma(i) = c(\rho(i))$. ρ is accepting if $C = \emptyset$ or $\max C$ is even, where $C = \mathrm{Inf}(c(\rho))$.

The language of $\mathcal{A}$, written $L(\mathcal{A}) \subseteq \Sigma^\infty$ is defined as the set of all ∞-words over Σ which have an accepting run on $\mathcal{A}$. We define $L_*(\mathcal{A}) = L(\mathcal{A}) \cap \Sigma^*$ and $L_\omega(\mathcal{A}) = L(\mathcal{A}) \cap \Sigma^\omega$.
$\mathcal{A}$ *recognizes* or *accepts* a language $L \subseteq \Sigma^\infty$ if $L(\mathcal{A}) = L$.

Every ω-regular language can be recognized by a standard parity automaton, even a deterministic one. (see [**?**]). The other two models are exactly as powerful.

Proposition 1.5.1. *Let $U \subseteq \Sigma^\omega$ be an ω-language. The following statements are equivalent:*

- *U is regular.*
- *There is a standard PA $\mathcal{A}$ with $L(\mathcal{A}) = U$.*
- *There is a transition PA $\mathcal{A}$ with $L_\omega(\mathcal{A}) = U$.*
- *There is an expression PA $\mathcal{A}$ with $L_\omega(\mathcal{A}) = U$.*

Proposition 1.5.2. *Let $L \subseteq \Sigma^\infty$ be an ∞-language. The following statements are equivalent:*

- *L is regular.*
- *There is a transition PA $\mathcal{A}$ with $L(\mathcal{A}) = L$.*
- *There is an expression PA $\mathcal{A}$ with $L(\mathcal{A}) = L$.*

Most of the constructions are trivial and require at most $|Q|$ many additional states. The only step not as clear is the question on how to transform an expression PA to one of the other models. That will be part of a later chapter, when we discuss the conversion from automata to regular expressions.

Definition 1.5.4. Let $\mathcal{A} = (Q, \Sigma, q_0, \delta, c, F)$ and $\mathcal{A}' = (Q', \Sigma, q_0', \delta', c', F')$ be transition parity automata. A function $f : Q \to Q'$ is an *isomorphism* from $\mathcal{A}$ to $\mathcal{A}'$, if

- f is bijective.
- $f(q_0) = q_0'$.
- For all $q \in Q$, for all $a \in \Sigma$: $\delta'(f(q), a) = \{f(p) \mid p \in \delta(q, a))\}$.
- For all $q \in Q$: $q \in F$ if and only if $f(q) \in F'$.
- There is a bijective, monotone function $n : \mathbb{Z} \to \mathbb{Z}$ (i.e. $x < y$ implies $n(x) < n(y)$) for which $n(x)$ is even if and only if x is even, such that for all $q \in Q$ and $p \in \text{succ}(q)$: $c(p, q) = n(c(f(p), f(q)))$. (In particular, this can be satisfied by the identity function.)

If an isomorphism from $\mathcal{A}$ to $\mathcal{A}'$ exists, $\mathcal{A}$ and $\mathcal{A}'$ are *isomorphic* ($\mathcal{A} \cong \mathcal{A}'$).

Proposition 1.5.3. *Let $\mathcal{A}$ and $\mathcal{A}'$ be transition parity automata. If $\mathcal{A} \cong \mathcal{A}'$, then $L(\mathcal{A}) = L(\mathcal{A}')$.*

Proof. $\mathcal{A} \cong \mathcal{A}'$, so let $f : Q \to Q'$ be an isomorphism from $\mathcal{A}$ to $\mathcal{A}'$. It is clear that f^{-1} is also an isomorphism from $\mathcal{A}'$ to $\mathcal{A}$ and therefore, it suffices to show that $L(\mathcal{A}) \subseteq L(\mathcal{A}')$. Let $w \in L(\mathcal{A})$ and let ρ be the run of $\mathcal{A}$ on w.

We first show that $\rho' : |\rho| \to Q', i \mapsto f(\rho(i))$ is a valid run of $\mathcal{A}'$ on w. We do so using induction over $|w|$. For $|w| = 0$, this is clear because $\rho'(0) = q_0' = f(q_0) = f(\rho(0))$. Otherwise, let $0 < |w|$. By the induction hypothesis, $\rho'[0, |w| - 1]$ is a valid run of $\mathcal{A}'$ on $w[0, |w| - 2]$. ρ is a valid run of $\mathcal{A}$ on w, so $\rho(|w|) \in \delta(\rho(|w| - 1), w(|w| - 1))$. The definition of an isomorphism then implies that $\delta'(f(\rho(|w| - 1)), w(|w| - 1)) = \delta'(\rho'(|w| - 1), w(|w| - 1)) = \{f(p) \mid p \in \delta(\rho(|w| - 1), w(|w| - 1))\}$. In particular, $f(\rho(|w|)) = \rho'(|w|) \in \delta'(\rho'(|w| - 1), w(|w| - 1))$, which is what had to be shown.

If $w \in \Sigma^*$, $\rho(|w|) \in F$. By definition, $\rho'(|w|) \in F'$, which means that $w \in L(\mathcal{A}')$.

If $w \in \Sigma^\omega$, let P be the priority set of ρ. $\max P$ must be even. By definition of the isomorphism, the priority set of ρ' is $P' = \{n(x) \mid x \in P\}$ for a fitting function n. n is monotone, so $(\max P) = \max P'$. Also, $\max P$ is even which means that $n(\max P) = \max P'$ must be even. Therefore, $w \in L(\mathcal{A}')$. □

1.6 Schewe

1.6.1 Schewe Automaton

Definition 1.6.1. Let $\mathcal{A} = (Q, \Sigma, q_0, \delta, c)$ be a deterministic parity automaton. For $w \in \Sigma^* \cup \Sigma^\omega$ and $q \in Q$, we define $\lambda_\mathcal{A}(q, w) \in \mathbb{N}^{1+|w|}$ as follows: Let $q_0 q_1 \cdots \in Q^{1+|w|}$ be the unique run of $\mathcal{A}$ on w. Then $\lambda_\mathcal{A}(q, w)(n) = c(q_n)$.

We define the **reachability order** $\preceq^\mathcal{A}_{\text{reach}} \subseteq Q \times Q$ as $p \preceq^\mathcal{A}_{\text{reach}} q$ iff q is reachable from p. ("p is closer to q_0 than q"). Note that $p \preceq^\mathcal{A}_{\text{reach}} q$ and $q \preceq^\mathcal{A}_{\text{reach}} p$ together mean that p and q reside in the same SCC.

Definition 1.6.2. Let $\mathcal{A} = (Q, \Sigma, q_0, \delta, c)$ be a DPA and let $\sim \subseteq Q \times Q$ be a congruence relation on $\mathcal{A}$. We define the **Schewe** automaton $\mathcal{S}(\mathcal{A}, \sim)$ as follows:

Let $\preceq \subseteq Q \times Q$ be a relation s.t.

- $\preceq$ is a total preorder.
- If q is reachable from p, then $p \preceq q$.
- If $p \simeq q$, then p and q are in the same SCC.

For each state q, let $[q]_\sim = \{p \in Q \mid q \sim p\}$ be its equivalence class of $\sim$ and let $Q/_\sim = \{[q]_\sim \mid q \in Q\}$ be the set of equivalence classes. For each such class $\mathfrak{c}$ we fix a representative $r_\mathfrak{c} \in \mathfrak{c}$ which is $\preceq$-maximal in its class.

The automaton is then almost the same as the original DPA, with only a few modifications. Namely, $\mathcal{S} = (Q, \Sigma, r_{[q_0]_\sim}, \delta_\mathcal{S}, c)$.

For each transition $\delta_\mathcal{S}(q, a)$, let $\delta(q, a) = p$. If $q \prec r_{[p]_\sim}$, then $\delta_\mathcal{S}(q, a) = r_{[p]_\sim}$. Otherwise, we keep $\delta_\mathcal{S}(q, a) = p$. In other words, every time a transition leaves the SCC, it skips to the representative which lies as "deep" inside the automaton as possible.

Note that the choice of $\preceq$ and the representatives allow some degree of freedom when construction the automaton, so in general there are multiple Schewe automata for each DPA. The relevant properties that we consider hold for all of them, so we just fix an arbitrary choice of all Schewe automata and call this **the** Schewe automaton.

Lemma 1.6.1. *For a given $\mathcal{A}$ and $\sim$, the Schewe automaton $\mathcal{S}$ can be computed in $\mathcal{O}(|\mathcal{A}|)$.*

Proof. The only interesting part is the computation of the total preorder $\preceq$. Once that is given, we use one loop over all states to find a $\preceq$-maximal representative for each $\sim$-class and another loop over all transitions to redirect target states to their representative.

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. □

1.6.2 Schewe automaton structure

Lemma 1.6.2. *Let $\mathcal{A}$ be a DPA and $\sim$ be a congruence relation. Let ρ be a run on α starting at a reachable state in $\mathcal{S}(\mathcal{A}, \sim)$. Then for all i, $\rho(i) \preceq \rho(i+1)$.*

Furthermore, we have $\rho(i) \prec \rho(i+1)$ if and only if $\rho(i) \prec r_{[\delta_{\mathcal{A}}(\rho(i),\alpha(i))]_\sim}$.

Proof. Let i be an arbitrary index of the run. If $\rho(i)$ to $\rho(i+1)$ is also a transition in $\mathcal{A}$, then $\rho(i+1)$ is reachable from $\rho(i)$ in $\mathcal{A}$ and hence $\rho(i) \preceq \rho(i+1)$ by definition of the preorder. Otherwise the transition used was redirected in the construction. The way the redirection is defined, this implies $\rho(i) \prec \rho(i+1)$.

We move on to the second part of the lemma. If $\rho(i) \prec r_{[\delta_{\mathcal{A}}(\rho(i),\alpha(i))]_\sim}$, then the transition is redirected to $\rho(i+1) = r_{[\delta_{\mathcal{A}}(\rho(i),\alpha(i))]_\sim}$ and the statement holds.

For the other direction, let $\rho(i) \prec \rho(i+1)$ and assume towards a contradiction that $\rho(i) \not\prec r_{[\delta_{\mathcal{A}}(\rho(i),\alpha(i))]_\sim}$. This means that the transition was not redirected and $\rho(i+1) = \delta_{\mathcal{A}}(\rho(i), \alpha(i))$. Since $\preceq$ is total, we have $r_{[\delta_{\mathcal{A}}(\rho(i),\alpha(i))]_\sim} = r_{[\rho(i+1)]_\sim} \preceq \rho(i) \prec \rho(i+1)$ which contradicts the $\preceq$-maximality of representatives. □

Lemma 1.6.3. *Let $\mathcal{A}$ be a DPA and $\sim$ be a congruence relation. Let p and q be two reachable states in $\mathcal{S}(\mathcal{A}, \sim)$. If $p \sim q$, then p and q lie in the same SCC.*

Proof. It suffices to restrict ourselves to $q = r_{[q]_\sim} = r_{[p]_\sim}$. If we can prove the Lemma for this case, then the general statement follows as every state in $[q]_\sim$ must lie in the same SCC as q.

Let $p_0 \cdots p_n$ be a minimal run of $\mathcal{S}$ that reaches p. By Lemma **??**, we have $p_0 \preceq \cdots \preceq p_n$. Whenever $p_i \prec p_{i+1}$, a redirected transition to the representative $r_{[p_{i+1}]_\sim} = p_{i+1}$ is taken.

Let k be the first position after which no redirected transition is taken anymore. For the first case, assume that $k < n$. Then $p_i \simeq r_{[p_{i+1}]_\sim}$ for all $i \geq k$. In particular, $p_{n-1} \simeq q$. Since $p_{n-1} \preceq p_n$, we also have $q \preceq p_n$. The representatives are chosen $\preceq$-maximal in their $\sim$-class, so $q \simeq p_n$.

The second case is $k = n$. In that case, the transition from p_{n-1} to p_n is redirected and $p_n = r_{[p_n]_\sim} = q$. □

Lemma 1.6.4. *Let $\rho \in Q^\omega$ be an infinite run in $\mathcal{S}$ starting at a reachable state. Then ρ has a suffix that is a run in $\mathcal{A}$.*

Proof. We show that only finitely often a redirected transition is used in ρ. Then, from some point on, only transitions that also exist in $\mathcal{A}$ are used. The suffix starting at this point is the run that we are looking for.

Let $\rho = p_0 p_1 \cdots$. By Lemma **??**, we have $p_i \preceq p_{i+1}$ for all i and $p_i \prec p_{i+1}$ whenever a redirected transition is taken. As Q is finite, we can only move up in the order finitely often. This proves our claim. □

Lemma 1.6.5. *For all $q \in Q$ and $a \in \Sigma$, $\delta_{\mathcal{A}}(q, a) \sim \delta_{\mathcal{S}(\mathcal{A},\sim)}(q, a)$.*

Proof. If $\delta_{\mathcal{A}}(q, a) = \delta_{\mathcal{S}(\mathcal{A},\sim)}(q, a)$, the statement is clear. Otherwise, the transition was redirected and $\delta_{\mathcal{S}(\mathcal{A},\sim)}(q, a) = r_{[\delta_{\mathcal{A}}(q,a)]_\sim}$ which also shows the $\sim$-equivalence of the two states. □

1.6.3 Special cases of the Schewe automaton

Definition 1.6.3. Two DPAs $\mathcal{A}$ and $\mathcal{B}$ are **priority almost-equivalent**, if for all words $\alpha \in \Sigma^\omega$, $\lambda_\mathcal{A}(q_0^\mathcal{A}, \alpha)$ and $\lambda_\mathcal{B}(q_0^\mathcal{B}, \alpha)$ differ in only finitely many positions. We call two states $p, q \in Q$ of $\mathcal{A}$ priority almost-equivalent, $\mathcal{A}_q$ and $\mathcal{A}_p$ are priority almost-equivalent, where $\mathcal{A}_q$ behaves like $\mathcal{A}$ with initial state q.

Lemma 1.6.6. *Priority almost-equivalence is a congruence relation.*

Proof. Obvious. □

Lemma 1.6.7. *Priority almost-equivalence implies language equivalence.*

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 β but infinitely often in a. Hence, α and β differ at infinitely many positions where a occurs in α. That would mean w is a witness that the two automata are not priority almost-equivalent, contradicting our assumption. □

Lemma 1.6.8. *Let $\mathcal{A}$ be a DPA and $\sim$ be a congruence relation that implies language equivalence. Then $\mathcal{A}$ and $\mathcal{S} := \mathcal{S}(\mathcal{A}, \sim)$ are language equivalent.*

Proof. Let $\alpha \in \Sigma^\omega$ be a word and let ρ be the run of $\mathcal{S}$ on α. By Lemma **??**, ρ has a suffix π which is a run segment of $\mathcal{A}$ on some suffix β of α. The acceptance condition of DPAs is prefix independent, so $\alpha \in L(\mathcal{S})$ iff ρ is an accepting run iff π is an accepting run. Since the priorities do not change during the construction, π is accepting in $\mathcal{S}$ iff it is accepting in $\mathcal{A}$.

Let $w \in \Sigma^*$ be the prefix of α with $\alpha = w\beta$. By Lemma **??**, we know that $\delta^*_\mathcal{A}(q_0, w) \sim \delta^*_\mathcal{S}(q_0, w)$. Since every state is $\sim$-equivalent to its representative and $\sim$ is a congruence relation, we also know $\delta^*_\mathcal{S}(q_0, w) \sim \delta^*_\mathcal{S}(r_{[q_0]_\sim}, w)$. From $\delta^*_\mathcal{S}(r_{[q_0]_\sim}, w)$, the run π accepts β iff $\alpha \in L(\mathcal{S})$. As $\sim$ implies language equivalence, the same must hold for $\delta^*_\mathcal{A}(q_0, w)$. Therefore, $\alpha \in L(\mathcal{A})$ iff $\alpha \in L(\mathcal{S})$. □

Lemma 1.6.9. *Let $\mathcal{A}$ be a DPA and $\sim$ be the relation of priority almost-equivalence. Let $\mathcal{S}'$ be the Moore-minimization of $\mathcal{S}(\mathcal{A}, \sim)$. There is no smaller DPA than $\mathcal{S}'$ that is priority almost-equivalent to $\mathcal{A}$.*

Proof. Let $\mathcal{B}$ be a DPA that is smaller than $\mathcal{S}'$. Our goal is to construct a word on which their priorities differ infinitely often.

First of all, $\mathcal{B}$ must have "the same" $\sim$-equivalence classes as $\mathcal{S}'$, i.e. for all states p in $\mathcal{S}'$, there is a state q in $\mathcal{B}$ s.t. $\mathcal{B}_q$ and $\mathcal{S}'_p$ are priority almost-equivalent. If this would not be the case, there is a p for which no such q exists. As $\mathcal{S}'$ was minimized, p is reachable by some word w. Whatever state q is reached in $\mathcal{B}$ via w, $\mathcal{B}_q$ and $\mathcal{S}'_p$ are not priority almost-equivalent and there is some witness α for that. Hence, $w\alpha$ is a witness that $\mathcal{B}$ and $\mathcal{S}'$ are not priority almost-equivalent.

We define f as a function that maps every $\sim$-equivalence class in $\mathcal{S}'$ to its respective class in $\mathcal{B}$. $\mathcal{B}$ has at least as many $\sim$-equivalence classes but less states than $\mathcal{S}'$, so there is a class $\mathfrak{c}$ in $\mathcal{S}'$ such that $f(\mathfrak{c})$ contains less elements than $\mathfrak{c}$.

By Lemma **??**, there is a unique SCC C in $\mathcal{S}'$ in which all states in $\mathfrak{c}$ are contained. The same must be true for some SCC D in $\mathcal{B}$ for the states in $f(\mathfrak{c})$. Otherwise we could apply the Schewe automaton construction to $\mathcal{B}$ which does not increase the number of states, is priority almost-equivalent to $\mathcal{B}$, and has this property.

C and D must be non-trivial SCCs. If C would be trivial, then $\mathfrak{c}$ would only contain one element and $f(\mathfrak{c})$ would be empty. Hence, one can force multiple visits to a state in $\mathfrak{c}$ in $\mathcal{S}'$. If D would be trivial, this would not be possible and we could again find a seperating witness of the two automata.

We claim: There is a state $p \in \mathfrak{c}$ s.t. for all $q \in f(\mathfrak{c})$, there is a word $w_q \in \Sigma^*$ that is a witness for non-Moore equivalence of $\mathcal{B}_q$ and $\mathcal{S}'_r$ and $\mathcal{S}'$ does not leave C when reading w_q from p. Assume the opposite, i.e. for all $p \in \mathfrak{c}$, there is a $q_p \in f(\mathfrak{c})$ which does not satisfy said property.

As $|\mathfrak{c}| < |f(\mathfrak{c})|$, there are two states p_1 and p_2 such that $q_{p_1} = q_{p_2} =: q$. For both $i \in \{1, 2\}$ and for each word $w \in \Sigma^*$, we have $\lambda_{\mathcal{B}}(q, w) = \lambda_{\mathcal{S}'}(p_i, w)$ or $\mathcal{S}'$ leaves the SCC C when reading w from p_i. If for both i and all words w the first case would apply, then p_1 and p_2 would be Moore-equivalent and would have been merged in the minimization process of $\mathcal{S}'$. Hence, there are i (which we assume to be $i = 1$ wlog) and w such that $\lambda_{\mathcal{B}}(q, w) \neq \lambda_{\mathcal{S}'}(p_i, w)$ but $\mathcal{S}'_{p_i}$ leaves C when reading w. Let this w be minimal in length.

Let ρ and π be the runs of $\mathcal{S}'$ from p_1 and p_2 via w. At some position k, ρ leaves C. By Lemma **??**, this means that a redirected transition was taken to $\rho(k+1) = r_{[\rho(k+1)]_\sim}$. $\sim$ is a congruence relation and $p_1 \sim p_2$, so $\pi(k+1) \sim \rho(k+1)$. As $p_1 \simeq p_2$ and $p_1 \prec r_{[\rho(k+1)]_\sim}$, we also have $p_2 \prec r_{[\rho(k+1)]_\sim}$ and $\pi(k+1) = \rho(k+1)$. As w was chosen minimal in length, the priorities of the runs are equal everywhere except for $\delta^*_{\mathcal{S}'}(p_1, w)$ and $\delta^*_{\mathcal{S}'}(p_2, w)$. However, the runs converge at $k+1$ which means that they visit the same states from that point on. In particular, $\delta^*_{\mathcal{S}'}(p_1, w) = \delta^*_{\mathcal{S}'}(p_2, w)$.

We have thus proven that the described state $p \in \mathfrak{c}$ and the words $w_q \in \Sigma^*$ exist. We can use these to finally construct our witness α. We define a sequence $(\alpha_n)_{n\in\mathbb{N}} \in \Sigma^*$ such that on α_n, the runs of $\mathcal{S}'$ and $\mathcal{B}$ differ at least n times in priority. Then $\alpha := \bigcup_n \alpha_n$ satisfies our requirements. Furthermore, we make sure that after reading α_n, $\mathcal{S}'$ always stops in p.

Every state in $\mathcal{S}'$ is reachable, so let α_0 be a word that reaches p from the initial state. Now assume that α_n was already defined. Let $q = \delta^*_{\mathcal{B}}(q_0^{\mathcal{B}}, \alpha_n)$. As $p \in \mathfrak{c}$, we can use the same argument as earlier to find $q \in f(\mathfrak{c})$. There is a suitable word w_q that is a witness for Moore non-equivalence while staying in C. As we stay in C, there is also a word u such that $\delta^*_{\mathcal{S}'}(p, w_q u) = p$. We set $\alpha_{n+1} := \alpha_n w_q u$. By choice of w_q, there is a position during the segment on $w_q u$ at which runs of $\mathcal{S}'$ and $\mathcal{B}$ differ in priority. By induction, α_{n+1} satisfies all our required properties. □

Definition 1.6.4. Let $\mathcal{A} = (Q_1, \Sigma, q_0^1, \delta_1, c_1)$ and $\mathcal{B} = (Q_2, \Sigma, q_0^2, \delta_2, c_2)$ be DPAs. We define the deterministic Büchi automaton $\mathcal{A} \intercal \mathcal{B} = (Q_1 \times Q_2, \Sigma, (q_0^1, q_0^2), \delta_\intercal, F_\intercal)$ with $\delta_\intercal((q_1, q_2), a) = (\delta_1(q_1, a), \delta_2(q_2, a))$. The transition structure is a common product automaton.

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

Lemma 1.6.10. *Let $\mathcal{A} = (Q_1, \Sigma, q_0^1, \delta_1, c_1)$ and $\mathcal{B} = (Q_2, \Sigma, q_0^2, \delta_2, c_2)$ be DPAs. $\mathcal{A}$ and $\mathcal{B}$ are priority almost-equivalent iff $L(\mathcal{A} \intercal \mathcal{B}) = \emptyset$.*

Proof. For the first direction of implication, let $L(\mathcal{A} \intercal \mathcal{B}) \neq \emptyset$, so there is a word α accepted by that automaton. Let $(p_0, q_0)(p_1, q_1) \cdots$ be the accepting run on α. Then $p_0 p_1 \cdots$ and $q_0 q_1 \cdots$ are the runs of $\mathcal{A}$ and $\mathcal{B}$ on α 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, α is a witness for $\mathcal{A}$ and $\mathcal{B}$ being not priority almost-equivalent.

For the second direction, let $\mathcal{A}$ and $\mathcal{B}$ be not priority almost-equivalent, so there is a witness α 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. □

Lemma 1.6.11. *The priority almost-equivalence of a DPA $\mathcal{A}$ can be computed in $\mathcal{O}(|\mathcal{A}|^2)$.*

Proof. The definition of $\mathcal{A} \intercal \mathcal{A}$ is a straightforward construction and can be done in quadratic time. By Lemma **??**, we know that p and q are priority almost-equivalent iff the language of $(\mathcal{A} \intercal \mathcal{A})_{(p,q)}$ is empty.

The product automaton $\mathcal{A} \intercal \mathcal{A}$ itself can be computed in quadratic time. The SCCs of $\mathcal{A} \intercal \mathcal{A}$ can be computed in linear time in said automaton, i.e. in $\mathcal{O}(|\mathcal{A}|^2)$.

The language from a given starting state is non-empty iff a goal SCC is reachable, i.e. an SCC that contains an accepting state and is non-trivial. It is a rather simple procedure to collect these SCCs and as described in the proof of Lemma **??**, we can merge all of them into a single state $\hat{q}$ in $\mathcal{O}(|\mathcal{A}^2|)$.

All that remains to be done is to find all states from which $\hat{q}$ is unreachable. This is easily done by using the transposed automaton of $\mathcal{A} \intercal \mathcal{A}$, which has all transition edges inverted, and performing a DFS starting in $\hat{q}$. States which are **not** found here correspond to priority almost-equivalent pairs. □

1.6.4 Efficiency

So far we used this section to define the Schewe construction and to establish its theoretical properties, namely that the new automaton is minimal in a certain way and that it can be computed in quadratic time. The rest of this section is used to present and analyze results of experiments done on the construction.

The algorithm was implemented in C++ and executed for different inputs, which were determined by the following properties:

1. Source of the automaton, which is one of three.
   - Use the *Spot* library to generate a random DPA.
   - Use the *Spot* library to generate a random NBA and *Spot* again to determinize it to a DPA.
   - Use the *Spot* library to generate a random NBA and use *nbautils* library to determinize it.
2. Number of states.
3. Number of priorities.
4. Size of the input alphabet Σ; since the tools described work with vectors of atomic propositions as alphabets, this value always is a power of 2.
5. Number of SCCs.

Afterwards, the efficiency was analyzed in the form of (relative or absolute) number of states that were removed. Note that this analysis always includes a Moore-minimization after performing the Schewe construction, as the construction itself does not remove any states.

Definition 1.6.5. Let $\mathcal{A} = (Q, \Sigma, q_0, \delta, c)$ be a DPA. We define the **Moore-minimization** $\mathcal{B}$ as the parity automaton corresponding to the minimal Moore automaton of $\mathcal{A}$. That means it is the minimal automaton such that $\lambda_{\mathcal{A}}(\alpha) = \lambda_{\mathcal{B}}(\alpha)$ for all $\alpha \in \Sigma^\omega$.

More specifically, we define the congruence relation $\sim \subseteq Q \times Q$ by $p \sim q$ iff $\forall w \in \Sigma^* : \lambda_{\mathcal{A}}(p, w) = \lambda_{\mathcal{A}}(q, w)$. Then $\mathcal{A}'$ is constructed from $\mathcal{A}$ by removing unreachable states (from q_0). $\mathcal{B} = (Q/_\sim, \Sigma, [q_0]_\sim, \delta_{\mathcal{B}}, c_{\mathcal{B}})$ is the quotient automaton of $\mathcal{A}'/_\sim$ with $\delta_{\mathcal{B}}([q]_\sim, a) = [\delta(q, a)]_\sim$ and $c_{\mathcal{B}}([q]_\sim) = c(q)$.

Lemma 1.6.12. *For a given DPA $\mathcal{A}$, the Moore-minimization can be computed in $\mathcal{O}$.*

1.7 Fritz & Wilke

1.7.1 Delayed Simulation Game

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

Definition 1.7.1. 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 1.7.2. Let $\mathcal{A} = (Q, \Sigma, q_0, \delta, c)$ be a DPA. We define the *delayed simulation automaton* $\mathcal{A}_\mathrm{de}(p, q) = (Q_\mathrm{de}, \Sigma, (p, q, \gamma(c(p), c(q), \checkmark)), \delta_\mathrm{de}, F_\mathrm{de})$, 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 Σ.
- 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 γ is defined below.
- $\delta_\mathrm{de}((p, q, k), a) = (p', q', \gamma(c(p'), c(q'), k)$, where $p' = \delta(p, a)$, $q' = \delta(q, a)$, and γ is the same function as used in the initial state. The first two components behave like a regular product automaton.
- $F_\mathrm{de} = Q \times Q \times \{\checkmark\}$.

$\gamma : \mathbb{N} \times \mathbb{N} \times (\mathbb{N} \cup \{\checkmark\}) \to \mathbb{N} \cup \{\checkmark\}$ 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 1.7.3. 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 $L(\mathcal{A}_\mathrm{de}(p, q)) = \Sigma^\omega$. In that case we write $p \leq_\mathrm{de} q$. If also $q \leq_\mathrm{de} p$ holds, we write $p \equiv_\mathrm{de} q$.

$\equiv_{\text{de}}$ is a congruence relation.

Our overall goal is to use $\equiv_{\text{de}}$ 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 1.7.1. *γ 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 γ. If i is odd, $i \leq_\checkmark k$ and $j \preceq_{\text{p}} i$, 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 $j \preceq_{\text{p}} i$, 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 $k \leq_\checkmark k'$, $\gamma(i,j,k) \leq_\checkmark \gamma(i,j,k')$. □

Lemma 1.7.2. *Let $\mathcal{A}$ be a DPA and let $p, q \in Q$, $k \in \mathbb{N} \cup \{\checkmark\}$. If the run of $\mathcal{A}_{de}$ starting at (p,q,k) on some $\alpha \in \Sigma^\omega$ is accepting, then for all $k \leq_\checkmark k'$ also the run of $\mathcal{A}_{de}$ starting at (p,q,k') on α is accepting.*

Proof. Let ρ be the run starting at (p,q,k) and let ρ' 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 ρ' 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 **??**. □

Lemma 1.7.3. *Let $\mathcal{A}$ be a DPA and $\rho \in Q_{de}^\omega$ be a run of $\mathcal{A}_{\text{de}}$ on some word. Let $k \in (\mathbb{N} \cup \{\checkmark\})^\omega$ be the third component during ρ. For all i, $k(i+1) \leq_\checkmark k(i)$ or $k(i+1) = \checkmark$.*

Proof. Follows directly from the definition of γ. □

Lemma 1.7.4. *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 $\mathcal{A}_{de}(p,q)$ on α. If ρ is not accepting, there is a position n such that*

- *$Occ(\{p_i \mid i \geq n\}) = Inf(\{p_i \mid i \in \mathbb{N}\})$,*
- *$Occ(\{q_i \mid i \geq n\}) = Inf(\{q_i \mid i \in \mathbb{N}\})$,*
- *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 ρ do not change anymore.

Proof. The first two requirements are clear. Since states not in $\text{Inf}(\{p_i \mid i \in \mathbb{N}\})$ 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 **??**, 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. □

Lemma 1.7.5. *Let $\mathcal{A}$ be a DPA with two states $p, q \in Q$. Let $\alpha \in \Sigma^\omega$ be an ω-word and let ρ_p and ρ_q be the respective runs of $\mathcal{A}$ on α 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 α. Let n_0 be a position as described in Lemma **??** (for ρ).

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 $k_{13}(m) = \checkmark$. □

Lemma 1.7.6. *Let $\mathcal{A}$ be a DPA. Then $\leq_{de}$ 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 α 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 ρ_{12}, ρ_{23}, and ρ_{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 α. Then ρ_{12} and ρ_{23} are accepting, whereas ρ_{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 ρ_{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 **??** (for ρ_{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 **??** 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 **??**). 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, ρ_{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, ρ_{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 1.7.7. *Let $\mathcal{A}$ be a DPA. Then $\equiv_{de}$ is a congruence relation.*

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

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

We want to mention here that $\equiv_{\mathrm{de}}$ 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.

Corollary 1.7.8. *Let $\mathcal{A}$ be a DPA and $\equiv_{de}$ the corresponding delayed simulation-relation. The quotient automaton $\mathcal{A}/_{\equiv_{de}}$ is well-defined and deterministic.*

Alternative characterization

Having described several properties of $\leq_{\mathrm{de}}$ and $\equiv_{\mathrm{de}}$ in the previous part, we briefly want to give an alternative description in corollary **??** 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 1.7.9. *Let $\mathcal{A}$ be a DPA with states $p, q \in Q$. Then $p \preceq_{de} q$ 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 $p \not\preceq_{\mathrm{de}} q$, so there is a word $\alpha \notin L(\mathcal{A}_{\mathrm{de}}(p, q))$ with the run of $\mathcal{A}_{\mathrm{de}}(p, q)$ being $(p_0, q_0, k_0)(p_1, q_1, k_1)\ldots$. By Lemma **??**, 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 β 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 γ 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 $p \not\preceq_{\mathrm{de}} q$.

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 $\mathcal{A}_{\mathrm{de}}(p, q)$ 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 $\alpha \notin L(\mathcal{A}_{\mathrm{de}}(p, q))$.

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 γ, $c(q_{j+1}) \leq k_j$ 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 β. 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 $\preceq_{\mathrm{de}}$ seems arbitrary, it allows for an easier definition of $\equiv_{\mathrm{de}}$ as is seen in the following statement.

Corollary 1.7.10. *Let $\mathcal{A}$ be a DPA with states $p, q \in Q$. Then $p \equiv_{de} q$ 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 $\min\{c(p'), c(q')\}$.

Correctness of the quotient

The quotient automaton itself is used "only" for state space reduction. The main point of delayed simulation is that the priorities of equivalent states can be made equivalent.

Theorem 1.7.11. *Let $\mathcal{A} = (Q, \Sigma, q_0, \delta, c)$ be a DPA. Let $\sim \subseteq Q \times Q$ be a congruence relation such that $p \sim q$ implies $c(p) = c(q)$. Then $L(\mathcal{A}) = L(\mathcal{A}/\sim)$.*

Proof. Since $\mathcal{A}$ is deterministic and $\sim$ is a congruence relation, $\mathcal{A}/\sim = (Q_\sim, \Sigma, [q_0]_\sim, \delta_\sim, c_\sim)$ is deterministic as well. Let $\alpha \in \Sigma^\omega$ be a word and let π and ρ be the runs of $\mathcal{A}$ and $\mathcal{A}/\sim$.

For each $i \in \mathbb{N}$, we have $\rho(i) = [\pi(i)]_\sim$ and $c_\sim(\rho(i)) = c(\pi(i))$. Thus, $\mathrm{Inf}(c(\pi)) = \mathrm{Inf}(c(\rho))$ and π is accepting iff ρ is accepting. □

Lemma 1.7.12. *Let $\mathcal{A}$ be a DPA and let π and ρ be runs of $\mathcal{A}$ on the same word but starting at different states. If $\pi(0) \equiv_{\text{de}} \rho(0)$, then* $\min Occ(c(\pi)) = \min Occ(c(\rho))$.

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

If k is even, let σ be the run of $\mathcal{A}_{\text{de}}(\pi(0), \rho(0))$ on α. Let n 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 $c(\rho(n)) \not\geq_p c(\pi(n))$. 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. The rest follows from the definition of γ.

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

Theorem 1.7.13. *Let $\mathcal{A} = (Q, \Sigma, q_0, \delta, c)$ be a DPA and let $p, q \in Q$ with $p \equiv_{de} q$ and $c(p) < c(q)$. Define $\mathcal{A}' = (Q, \Sigma, q_0, \delta, c')$ with $c'(s) = \begin{cases} c(p) & \text{if } s = q \\ c(s) & \text{else} \end{cases}$. Then $L(\mathcal{A}) = L(\mathcal{A}')$.*

Proof. 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 $L(\mathcal{A}) \subseteq L(\mathcal{A}')$. For the other direction, assume there is a $\alpha \in L(\mathcal{A}') \backslash L(\mathcal{A})$, so the respective run $\rho \in Q^\omega$ is accepting in $\mathcal{A}'$ but not in $\mathcal{A}$.

For this to be true, ρ 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^ω and the run π_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 π_p on w^ω starting at p though, we find that this run sees priority $c(p) = c'(q)$ at the very beginning. This contradicts Lemma **??**, as $p \equiv_{\text{de}} q$. Thus, the described α 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. □

Corollary 1.7.14. *For a DPA $\mathcal{A}$, the quotient automaton $\mathcal{A}/_{\equiv_{de}}$ is a DPA that recognizes the same language.*

1.7.2 Using delayed simulation for APAs

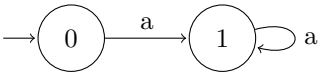

Figure 1.1: Example automaton in which the states could be merged but delayed simulation separates them.

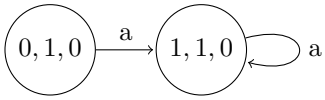

Figure 1.2: Example automaton in which the states could be merged but delayed simulation separates them.

1.7.3 Alternative computation

As we have seen, using delayed simulation to build a quotient automaton delivers good results in the number of removed states. The downside is the computation time which is much higher than that of our approach in section **??**. In the following we will consider alternations to the delayed simulation algorithm with the goal to increase the number of removed states or to reduce computation time.

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 **??**; the two states could be merged into one, but they are not $\equiv_{\mathrm{de}}$-equivalent as can be seen in the delayed simulation automaton in figure **??**.

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.

Definition 1.7.4. Let $\mathcal{A} = (Q, \Sigma, q_0, \delta, c)$ be a DPA. We define the *delayed simulation automaton with SCC resets* $\mathcal{A}_{\mathrm{deR}}(p, q) = (Q_{\mathrm{de}}, \Sigma, (p, q, \gamma(c(p), c(q), \checkmark), \delta_{\mathrm{deR}}, F_{\mathrm{de}})$ with $\delta_{\mathrm{deR}}((p, q, k), a) = \delta_{\mathrm{de}}((p, q, \mathrm{reset}(p, q, k, a)), a)$. Except for the addition of the reset function, this automaton is the same as $\mathcal{A}_{\mathrm{de}}$.

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

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

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

Lemma 1.7.16. *Let $\mathcal{A}$ be a DPA with two states p and q. If $p \leq_{de} q$, then $p \leq_{deR} q$.*

Proof. Consider the two simulation automata $\mathcal{A}_{\mathrm{de}}(p,q)$ and $\mathcal{A}_{\mathrm{deR}}(p,q)$ 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 α. We claim that $k_i \leq_{\checkmark} l_i$ at every position. Then, since $L(\mathcal{A}_{\mathrm{de}}(p,q)) = \Sigma^\omega$, 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 **??**. Otherwise, $l_{i+1} = \checkmark \geq_{\checkmark} k_{i+1}$. □

The real difference comes up when looking at theorem **??**. If we consider again the example given in figure **??**, if that theorem would hold the same for $\equiv_{\mathrm{deR}}$ 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 1.7.17. *Let $\mathcal{A}$ be a DPA and let π and ρ be runs of $\mathcal{A}$ on the same word but starting at different states. Let off_π and off_ρ be the positions after which π and ρ 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 $\min \mathit{Occ}(c(\pi[\mathit{off},\omega])) = \min \mathit{Occ}(c(\rho[\mathit{off},\omega]))$.*

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

If k is even, let σ be the run of $\mathcal{A}_{\mathrm{de}}(\pi(0),\rho(0))$ on α. Let $n \geq \mathrm{off}$ 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 $c(\rho(n)) \not\leq_p c(\pi(n))$. 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 γ.

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. □

Theorem 1.7.18. *Let $\mathcal{A} = (Q,\Sigma,q_0,\delta,c)$ be a DPA and let $p,q \in Q$ with $p \equiv_{de} q$ 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 $L(\mathcal{A}) = L(\mathcal{A}')$.*

Proof. 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 $L(\mathcal{A}) \subseteq L(\mathcal{A}')$. For the other direction, assume there is a $\alpha \in L(\mathcal{A}')\backslash L(\mathcal{A})$, so the respective run $\rho \in Q^\omega$ is accepting in $\mathcal{A}'$ but not in $\mathcal{A}$.

For this to be true, ρ 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^ω and the run π_q of $\mathcal{A}$ on said word starting in q. With the argument above, we know that the minimal priority occuring in $c(\pi)$ is greater than $c'(q)$. If we take the run π_p on w^ω starting at p though, we find that this run sees priority $c(p) = c'(q)$ at the very beginning. This contradicts Lemma **??**, as $p \equiv_{\mathrm{de}} q$. Thus, the described α 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. □

Delayed simulation with normalized priorities

Additional properties of delayed simulation can be found if we look at only a subclass of all DPAs. In particular, we use automata with normalized priorities here.

Definition 1.7.5. Let $\mathcal{A} = (Q, \Sigma, \delta, q_0, c)$ be a parity automaton. We call c *normalized* if for every state $q \in Q$ that does not lie in a trivial SCC and all priorities $k \leq c(q)$, there is a path from q to q such that the lowest priority visited is k.

Algorithm **??** shows how an equivalent normalized priority function can be computed in $\mathcal{O}(nk)$, where n and k are the number of states and priorities respectively. The algorithm is a slight adaption of that presented in [], which is why we will not go into further details here and just refer to the original source.

Lemma 1.7.19. *Let $\mathcal{A}$ be a DPA with a normalized priority function and let p and q be states that do not lie in trivial SCCs. Then $p \equiv_{de} q$ if and only if $p \sim_M q$.*

Proof. The "if"-implication was shown in **??**. For the other direction, let $p \not\sim_M q$, so there is a word $w \in \Sigma^*$ such that $c(p') \neq c(q')$, where $p' = \delta^*(p, w)$ and $q' = \delta^*(q, w)$. Without loss of generality, assume $c(p') < c(q')$.

As c is normalized, there is a word u such that q' reaches again q' via u and sees only priorities greater or equal to $c(q')$. That means that on the path that is obtained from q' by reading u^ω, the priority $c(p')$ is never visited. By corollary **??**, that means $p \not\equiv_{\text{de}} q$. □

Iterated Moore equivalence

Our next approach differs greatly in its computation from the delayed simulation (or rather, it is not related at all to the delayed simulation automaton anymore) but will yield a result that is at least as good. As before, we focus on normalized DPAs here.

Definition 1.7.6. Let $\mathcal{A} = (Q, \Sigma, q_0, \delta, c)$ and $\mathcal{B} = (P, \Sigma, p_0, \varepsilon, d)$ be DPAs and $S \subseteq Q$. We say that $\mathcal{A}$ *prepends S to $\mathcal{B}$* if

- $q_0 \in S$
- $Q = P \dot\cup S$
- $\delta \restriction_{P \times \Sigma} = \varepsilon$
- $c \restriction_P = d$

We assume S to be an SCC in these use cases, i.e. from every $s \in S$, every other $s' \in S$ is reachable in $\mathcal{A}$.

Definition 1.7.7. Let $\mathcal{A}$ be a DPA with SCCs $\mathcal{S} \subseteq 2^Q$ without unreachable states. Let $\preceq \subseteq \mathcal{S} \times \mathcal{S}$ be a total preorder such that $S \preceq S'$ implies that S' is reachable from S. For the i-th element w.r.t. this order, we write S_i, i.e. $S_0 \prec S_1 \prec \cdots \prec S_{|\mathcal{S}|}$.

For every state q in $\mathcal{A}$, let $\text{SCC}(q)$ be the SCC of q and let $\text{SCCi}(q)$ be the index of that SCC, i.e. $\text{SCC}(q) = S_{\text{SCCi}(q)}$. Let $\preceq_Q \subseteq Q \times Q$ be a total preorder on the states such that $q \preceq_Q q'$ implies $\text{SCCi}(q) \leq \text{SCCi}(q')$.

We inductively define a sequence of automata $(\mathcal{B}_i)_{0 \leq i \leq |\mathcal{S}|}$. For every i, we write $\mathcal{B}_i = (Q_i, \Sigma, q_0^i, \delta_i, c_i)$.

Algorithm 1 Normalizing the priority function of a DPA.

```
1: function NORMALIZE(𝒜)
2:     c′ : Q → ℕ, q ↦ c(q)
3:     M(𝒜, c′)
4:     return c′
5: end function

6: function M(𝒜 ↾_P, c′)
7:     if P = ∅ then
8:         return 0
9:     end if
10:    min ← 0
11:    for SCC S in 𝒜 ↾_P do
12:        m := min c(S)  mod 2
13:        X := c^{-1}(m)
14:        for q ∈ X do
15:            c′(q) ← m
16:        end for
17:        S′ := S \ X
18:        m′ ←M(𝒜 ↾_{S′}, c′)
19:        if m′ even then
20:            if m even then
21:                δ := m
22:            else
23:                δ := m − 2
24:            end if
25:        else
26:            δ := m − 1
27:        end if
28:        for q ∈ S′ do
29:            c′(q) ← c′(q) − δ
30:        end for
31:        min ← min{min, m}
32:    end for
33:    return min
34: end function
```

- The state sets are defined as $Q_i = \bigcup_{j=i}^{|\mathcal{S}|} S_j$.
- The base case is $\mathcal{B}_{|\mathcal{S}|} = \mathcal{A} \restriction_{S_{|\mathcal{S}|}}$.
- Given that $\mathcal{B}_{i+1}$ is defined, let $\mathcal{B}'_i = (Q_i, \Sigma, q_0'^i, \delta'_i, c'_i)$ be a DPA that prepends S_i to $\mathcal{B}_{i+1}$ such that $\delta \restriction_{Q_i \times \Sigma} = \delta'_i$ and $c \restriction_{S_i} = c'_i \restriction_{S_i}$. If $i = 0$, we require $q_0'^i = q_0$.
- Let $q_0^i = q_0'^i$ and $\delta_i = \delta'_i$.
- Let M'_i be the Moore equivalence on $\mathcal{B}'_i$. If $S_i = \{q\}$ is a trivial SCC, q is not M'_i-equivalent to any other state, and there is a $p \in Q_{i+1}$ such that for all $a \in \Sigma$ $(\delta'_i(q,a), \delta'_i(p,a)) \in M'_i$, then let p_0 be $\preceq_Q$-maximal among those p and let $c_i(r) = \begin{cases} c_{i+1}(p_0) & \text{if } r = q \\ c_{i+1}(r) & \text{else} \end{cases}$. If any of the three conditions is false, simply set $c_i = c'_i$.

Let M_i be the Moore equivalence on $\mathcal{B}_i$. We define $\sim_{IM} := M_0$ and call this the *iterated Moore equivalence* of $\mathcal{A}$.

At first, this definition might seem complex and confusing when written down formally like this. More casually explained, we continuously add the SCCs of $\mathcal{A}$ starting from the "back". In addition to computing the usual Moore equivalence on our automaton, we also take trivial SCCs into special consideration; as their priority cannot appear infinitely often on any run, its value is effectively arbitrary. We can therefore perform extra steps to more liberally merge it with other states.

In the upcoming statements we prove the important properties of iterated Moore equivalence.

Lemma 1.7.20. *Let variables be as in definition* **??**. *For all i and all $p, q \in Q_{i+1}$, $(p,q) \in M'_i$ iff $(p,q) \in M_{i+1}$. For all i and all $p, q \in Q_{i+1}$, $(p,q) \in M_i$ iff $(p,q) \in M'_i$.*

Proof. □

Theorem 1.7.21. *Let variables be defined as in definition* **??** *and let $\sim_M$ be the Moore equivalence of $\mathcal{A}$. Then $\sim_M \subseteq \sim_{IM}$.*

Proof. Let p, q be two states with $p \sim_M q$ but $p \not\sim_{IM} q$ such that $\max\{\mathrm{SCCi}(p), \mathrm{SCCi}(q)\}$ is maximal among all possible pairs; if there are multiple pairs with the highest value, choose a pair which has maximal $\min\{\mathrm{SCCi}(p), \mathrm{SCCi}(q)\}$ as well.

Without loss of generality we can assume that $c_0(p) \neq c_0(q)$ as we can always find such a pair with the definition of Moore equivalence. Since $c(p) = c(q)$, at least one of the two states must have a different priority in c_0 compared to c. By symmetry, assume that state is q. $\{q\}$ must be a trivial SCC in $\mathcal{A}$. Let q' be the $\preceq_Q$-maximal state s.t. $(\delta(q,a), \delta(q',a)) \in M'_{\mathrm{SCCi}(q)}$ for all $a \in \Sigma$, that is the state whose priority was copied to q.

We can make the assumption that $\mathrm{SCCi}(q) > \mathrm{SCCi}(p)$: if $c(p) \neq c_0(\mathrm{SCCi}(p))$, then $\{p\}$ is a trivial SCC and the choice of the states is symmetric. Otherwise, if we would have $\mathrm{SCCi}(p) > \mathrm{SCCi}(q)$ and $c(p) = c_0(\mathrm{SCCi}(p))$, then $(p,q) \in M_{\mathrm{SCCi}(q)} \subseteq \sim_{IM}$.

Consider the case that $\mathrm{SCC}(p)$ is a non-trivial SCC, so there is a non-empty word w with $\delta^*(p,w) = p$. Because of the congruence property, we know $q \sim_M p \sim_M \delta^*(p,w) \sim_M \delta^*(q,w)$. As $\{q\}$ is a trivial SCC, $\mathrm{SCCi}(\delta^*(q,w)) > \mathrm{SCCi}(q)$. That means that every state that is $\sim_M$-equivalent to $\delta^*(q,w)$ must also be $\sim_{IM}$-equivalent to it; otherwise, the choice of (p,q) as a pair with maximal

value of SCCi(q) would be contradicted. In particular, $q \sim_{IM} \delta(q, w) \sim_{IM} p$, which breaks our initial assumption.

Finally, we look at the case that SCC(p) = $\{p\}$ is trivial. First, if $c(p) \neq c_0(p)$, then there must be a state p' s.t. $(\delta(p, a), \delta(p', a)) \in M'_{\text{SCCi}(p)}$ for all $a \in \Sigma$; consider the $\preceq_Q$-maximal state that satisfies this. We can show that p' must be the same as q' and therefore $c_0(p) = c_0(q') = c_0(q)$, which would be a contradiction. The only possibility for this to be false is that there is an $a \in \Sigma$ such that $(\delta(p, a), \delta(p', a)) \in M'_{\text{SCCi}(p)}$ but $(\delta(p, a), \delta(q', a)) \notin M'_{\text{SCCi}(p)}$. Remember that $(\delta(q, a), \delta(q', a)) \in M'_{\text{SCCi}(p)}$ and $\delta(q, a) \sim_M \delta(p, a)$. If now $(\delta(p, a), \delta(q', a)) \notin M'_{\text{SCCi}(p)}$ would hold, then $(\delta(p, a), \delta(q, a)) \notin M'_{\text{SCCi}(p)} \subseteq M_0$, so $\delta(p, a) \not\sim_{IM} \delta(q, a)$. Since $\{q\}$ is a trivial SCC, SCCi(q) < SCCi($\delta(p, a)$), meaning that the pair $(\delta(p, a), \delta(q, a))$ would contradict our choice of (p, q).

Second, if $c(p) = c_0(p)$, then there must be a state $p' \neq p$ s.t. $(p, p') \in M'_{\text{SCCi}(p)} \subseteq M_0$. Let that p' be $\preceq_Q$-maximal. SCC(p') cannot be a trivial SCC: note that $q \sim_{IM} q'$ and $p \sim_{IM} p'$. Furthermore, $\delta(p, a) \sim_{IM} \delta(q, a)$ for all a, as to not contradict the pair (p, q). Put together this means $\delta(p', a) \sim_{IM} \delta(q', a)$ for all a and therefore $c_0(p) = c_0(p') = c_0(q') = c_0(q)$.

Hence, SCC(p') is a non-trivial SCC. There is a non-empty word w s.t. $\delta^*(p', w) = p'$. □

Theorem 1.7.22. *For a DPA $\mathcal{A}$, $\mathcal{A}/_{\sim_{IM}}$ is also a DPA with $L(\mathcal{A}/_{\sim_{IM}}) = L(\mathcal{A})$.*

Proof. From the definition of the $\mathcal{B}_i$ sequence in the construction of $\sim_{IM}$, it becomes clear that $\mathcal{B}_0$ and $\mathcal{A}$ are isomorphic up to the priority function at some trivial SCCs. As those priorities are only seen finitely often anyway, they do not impact the acceptance of a word. □

Another nice and maybe surprising result is the relation of iterated Moore equivalence to delayed simulation.

Theorem 1.7.23. *Let $\mathcal{A}$ be a DPA. Then $\equiv_{de} \subseteq \sim_{IM}$.*

Proof. □

1.8 Congruence Path Refinement

In this section we present an algorithm that uses an existing congruence relation and refines it to the point where equivalent states can be "merged".

Definition 1.8.1. Let $\mathcal{A} = (Q, \Sigma, q_0, \delta, c)$ be a DPA and let $\equiv \subseteq Q \times Q$ be an equivalence relation on the state set. For every equivalence class $\kappa \subseteq Q$, let $r_\kappa \in \kappa$ be an arbitrary representative of that class. For a DPA $\mathcal{A}' = (Q', \Sigma, q_0', \delta', c')$, we say that $\mathcal{A}'$ is a *representative merge of* $\mathcal{A}$ *w.r.t.* $\equiv$ if it satisfies the following:

- $Q' = \{r_{[q]_\equiv} \subseteq Q \mid q \in Q\}$
- $q_0' = r_{[q_0]_\equiv}$
- For all $q \in Q'$ and $a \in \Sigma$: $\delta'(q, a) = r_{[\delta(q,a)]_\equiv}$
- $c' = c \restriction_{Q'}$

Definition 1.8.2. Let $\mathcal{A} = (Q, \Sigma, q_0, \delta, c)$ be a DPA and let $R \subseteq Q \times Q$ be a congruence relation on the state space. Let $\lambda \subseteq Q$ be an equivalence class of R. We define $L_{\lambda\hookleftarrow}$ as the set of words w such that for any $u \sqsubseteq w$, $(\delta(p, u), q) \in R$ iff $u \in \{\varepsilon, w\}$. In other words, the set contains all minimal words by which the automaton moves from λ to λ again.

Let $f_{\mathrm{PR}} : 2^{\lambda\times\lambda} \to 2^{\lambda\times\lambda}$ be a function such that $(p, q) \in f(X)$ iff for all $w \in L_{\lambda\hookleftarrow}$, $(\delta^*(p, w), \delta^*(q, w)) \in X$. Then let $X_0 \subseteq \lambda \times \lambda$ such that $(p, q) \in X_0$ iff for all $w \in L_{\lambda\hookleftarrow}$, $\min\{c(\delta^*(p, u)) \mid u \sqsubset w\} = \min\{c(\delta^*(q, u)) \mid u \sqsubset w\}$, i.e. the minimal priority when moving from p or q to λ again is the same.

Using both, we set $X_{i+1} = f_{\mathrm{PR}}(X_i)$. f_{PR} is monotone w.r.t. $\subseteq$, so there is an $X_n = X_{n+1}$ by Kleene's fixed point theorem. We define the *path refinement of* λ, called $\equiv^\lambda_{\mathrm{PR}}$, as

- For $p \in Q \setminus \lambda$, $p \equiv^\lambda_{\mathrm{PR}} q$ iff $p = q$.
- For $p, q \in \lambda$, $p \equiv^\lambda_{\mathrm{PR}} q$ iff $(p, q) \in X_n$.

Theorem 1.8.1. *Let $\mathcal{A} = (Q, \Sigma, q_0, \delta, c)$ be a DPA and let $R \subseteq Q \times Q$ be a congruence relation that implies language equivalence. Let $\mathcal{A}'$ be a representative merge of $\mathcal{A}$ w.r.t. $\equiv^\lambda_{PR}$ for some equivalence class λ of R. Then $L(\mathcal{A}) = L(\mathcal{A}')$.*

Proof. Let $\alpha \in \Sigma^\omega$ be a word with runs $\rho \in Q^\omega$ and $\rho' \in (Q')^\omega$ of $\mathcal{A}$ and $\mathcal{A}'$ respectively. Let $k_0, \dots \in \mathbb{N}$ be exactly those positions (in order) at which ρ reaches λ, and analogously $k_0', \dots$ for ρ'.

Claim 1: For every i, $k_i = k_i'$ and $\rho(k_i) \equiv^\lambda_{\mathrm{PR}} \rho'(k_i)$. For all $j < k_0$, we know that $\rho(j) = \rho'(j)$, as no redirected edge is taken. Thus, $\rho'(k_0) = r_{[\rho(k_0)]_{\equiv^\lambda_{\mathrm{PR}}}} \equiv^\lambda_{\mathrm{PR}} = \rho(k_0)$.

Now assume that the claim holds for all $i \leq n$. By definition, $w = \alpha[k_n, k_{n+1}] \in L_{\lambda\hookleftarrow}$ and therefore $\rho(k_{n+1}) = \delta^*(\rho(k_n), w) \equiv^\lambda_{\mathrm{PR}} \delta^*(\rho'(k_n), w) = \rho'(k_{n+1})$.

Claim 2: If λ only occurs finitely often in ρ and ρ', then ρ is accepting iff ρ' is accepting.

Let $k_n \in \mathbb{N}$ be the last position at which $\rho(k_n)$ and $\rho'(k_n)$ are in λ. From this point on, $\rho'[k_n, \omega]$ is also a valid run of $\mathcal{A}$ on $\alpha[k_n, \omega]$. $\rho(k_n), \rho'(k_n) \in \lambda$, so $(\rho(k_n), \rho'(k_n)) \in R$. As R implies language equivalence, reading $\alpha[k_n, \omega]$ from either state in $\mathcal{A}$ leads to the same acceptance status. This also means that $\rho'(k_n)$ has the same acceptance status as $\rho(k_n)$.

Claim 3: If λ occurs infinitely often in ρ and ρ', then ρ is accepting iff ρ' is accepting.

For each i, $\alpha[k_i, k_{i+1}] \in L_{\lambda\hookleftarrow}$ by choice of the k_i. Hence, $\min \mathrm{Occ}(c(\rho[k_i, k_{i+1}])) = \min \mathrm{Occ}(c'(\rho'[k_i, k_{i+1}]))$ follows directly from the definition of $\equiv^{\lambda}_{\mathrm{PR}}$. Extending that result gives us $\min \mathrm{Inf}(c(\rho)) = \min \mathrm{Inf}(c'(\rho'))$. □

Lemma 1.8.2. *Let $\mathcal{A} = (Q, \Sigma, q_0, \delta, c)$ be a DPA and let $R \subseteq Q \times Q$ be a congruence relation that implies language equivalence. Let $\mathcal{A}'$ be a representative merge of $\mathcal{A}$ w.r.t. $\equiv^{\lambda}_{PR}$ for some equivalence class λ of R. Then $R \restriction_{Q'}$ still is a congruence relation that implies language equivalence in $\mathcal{A}'$.*

Proof. □

1.8.1 Algorithmic Definition

The definition of path refinement that we introduced is useful for the proofs of correctness. It however does not provide one with a way to actually compute the relation. That is why we now provide an alternative definition that yields the same results but is more algorithmic in nature.

Definition 1.8.3. Let $\mathcal{A} = (Q, \Sigma, q_0, \delta, c)$ be a DPA and let $R \subseteq Q \times Q$ be a congruence relation. For each equivalence class λ of R, we define the *path refinement automaton* $\mathcal{G}^{R,\lambda}_{\mathrm{PR}}(p, q) = (Q_{\mathrm{PR}}, \Sigma, q^{p,q}_{0,\mathrm{PR}}, \delta^{\lambda}_{\mathrm{PR}}, F_{\mathrm{PR}})$, which is a DFA.

- $Q_{\mathrm{PR}} = (Q \times Q \times c(Q) \times \{<, >, =\}) \cup \{\bot\}$
- $q^{p,q}_{0,\mathrm{PR}} = (p, q, \eta_k(c(p), c(q), \checkmark), \eta_x(c(p), c(q), \checkmark, =))$
- $\delta^{\lambda}_{\mathrm{PR}}((p, q, k, x), a) = \begin{cases} (p', q', \eta_k(c(p'), c(q'), k), \eta_x(c(p'), c(q'), k, x)) & \text{if } p' \notin \lambda \\ q^{p',q'}_{0,\mathrm{PR}} & \text{if } p' \in \lambda \text{ and } (x == =) \\ \bot & \text{else} \end{cases}$
  where $p' = \delta(p, a)$ and $q' = \delta(q, a)$.
  $\eta_k(k_p, k_q, k) = \min_{\leq_\checkmark}\{k_p, k_q, k\}$
  $\eta_x(k_p, k_q, k, x) = \begin{cases} < & \text{if } (k_p <_\checkmark k_q \text{ and } k_p <_\checkmark k) \text{ or } (k < k_q \text{ and } (x = <)) \\ > & \text{if } (k_p >_\checkmark k_q \text{ and } k >_\checkmark k_q) \text{ or } (k_p > k \text{ and } (x = >)) \\ = & \text{else} \end{cases}$
- $F_{\mathrm{PR}} = Q_{\mathrm{PR}} \setminus \{\bot\}$

Lemma 1.8.3. *Let $\mathcal{A}$ be a DPA with a congruence relation R. Let λ be an equivalence class of R, $p, q \in \lambda$, and $w \in L_{\lambda\to\lambda}$. For every $v \sqsubset w$ and $\oplus \in \{<, >, =\}$, the fourth component of $(\delta^{\lambda}_{PR})^*(q_{0,PR}, v)$ is $\oplus$ if and only if $\min\{c(\delta^*(p, u)) \mid u \sqsubseteq v\} \oplus \min\{c(\delta^*(q, u)) \mid u \sqsubseteq v\}$.*

The proof of this Lemma is a very formal analysis of every case in the relations between the different priorities that occur and making sure that the definition of η_x covers these correctly. No great insight is gained, which is why we omit the proof at this point.

Theorem 1.8.4. *Let $\mathcal{A}$ be a DPA with a congruence relation R. Let λ be an equivalence class of R and $p, q \in \lambda$. Then $p \equiv^{R}_{PR} q$ iff $L(\mathcal{G}^{R,\lambda}_{PR}(p, q)) = \Sigma^*$.*

Proof. **If** Let $p \not\equiv^{R}_{\mathrm{PR}} q$. Similarly to the proof of Lemma **??**, we use the inductive definition of $R_\kappa \subseteq \equiv^{R}_{\mathrm{PR}}$ using f and the sets X_i here. Let m be the smallest index at which $(p, q) \notin X_m$. Let

$\rho = (p_i, q_i, k_i, x_i)_{0 \leq i \leq |w|}$ be the run of $\mathcal{G}_{\mathrm{PR}}^{R,\lambda}(p,q)$ on w. We prove that $\rho(|w|) = \bot$ and therefore ρ is not accepting by induction on m.

If $m = 0$, then $(p,q) \notin Y_\lambda$, meaning that there is a word w such that $\min\{c(\delta^*(p,u)) \mid u \sqsubset w\} \neq \min\{c(\delta^*(q,u)) \mid u \sqsubset w\}$. Without loss of generality, assume $\min\{c(\delta^*(p,u)) \mid u \sqsubset w\} < \min\{c(\delta^*(q,u)) \mid u \sqsubset w\}$. By Lemma 0.1.3, $x_{|w|-1} = <$. Furthermore, $\delta(p_{|w|-1}, w_{|w|-1}) \in \lambda$, as $w \in L_{\lambda\to\lambda}$. Thus, $\rho(|w|) = \bot$ and the run is rejecting.

Now consider $m + 1 > 1$. Since $(p,q) \in X_m \setminus f(X_m)$, there must be a word $w \in L_{\lambda\to\lambda}$ such that $(p',q') \notin X_m$, where $p' = \delta^*(p,w)$ and $q' = \delta^*(q,w)$. As $R_\kappa \subseteq X_m$, $(p',q') \notin R_\kappa$ and therefore $p' \not\equiv_{\mathrm{PR}}^R q'$. By induction, $w \notin L(\mathcal{G}_{\mathrm{PR}}^{R,\lambda}(p',q'))$; since that run is a suffix of ρ, ρ itself is also a rejecting run.

Only If Let $L(\mathcal{G}_{\mathrm{PR}}^{R,\lambda}(p,q)) \neq \Sigma^*$. Since ε is always accepted, there is a word $w \in \Sigma^+ \setminus L(\mathcal{G}_{\mathrm{PR}}^{R,\lambda}(p,q))$, meaning that $\delta_{\mathrm{PR}}^*(q_{0,\mathrm{PR}}, w) = \bot$. Split w into sub-words $w = u_1 \cdots u_m$ such that $u_1, \ldots, u_m \in L_{\lambda\to\lambda}$. Note that this partition is unique. We show $p \not\equiv_{\mathrm{PR}}^R q$ by induction on m. Let $\rho = (p_i, q_i, k_i, x_i)_{0 \leq i < |w|}$ be the run of $\mathcal{G}_{\mathrm{PR}}^{R,\lambda}(p,q)$ on w.

If $m = 1$, then $w \in L_{\lambda\to\lambda}$. Since $\rho(|w|) = \bot$, it must be true that $x_{|w|-1} \neq =$. Without loss of generality, assume $x_{|w|-1} = <$. By Lemma 0.1.3, $\min\{c(\delta^*(p,u)) \mid u \sqsubset w\} < \min\{c(\delta^*(q,u)) \mid u \sqsubset w\}$. Therefore, $p \not\equiv_{\mathrm{PR}}^R q$.

Now consider $m + 1 > 1$. Let $p' = \delta^*(p, u_1)$ and $q' = \delta^*(q, u_1)$. By induction on the word $u_2 \cdots u_m$, $p' \not\equiv_{\mathrm{PR}}^R q'$. Since $u_1 \in L_{\lambda\to\lambda}$, that also means $p \not\equiv_{\mathrm{PR}}^R q$. □

The differences between different $\mathcal{G}_{\mathrm{PR}}^{R,\lambda}$ for different λ are minor and the question whether the accepted language is universal boils down to a simple question of reachability. Thus, $\equiv_{\mathrm{PR}}^R$ can be computed in $\mathcal{O}(|\mathcal{G}_{\mathrm{PR}}^{R,\lambda}|)$ which is $\mathcal{O}(|Q|^2 \cdot |c(Q)|)$.