0.1 Fritz & Wilke

0.1.1 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 0.1.1. 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 0.1.2. 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 0.1.3. 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 0.1.1. *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 0.1.2. *Let $\mathcal{A}$ be a DPA and let $p, q \in Q$, $k \in \mathbb{N} \cup \{\checkmark\}$. If the run of $\mathcal{A}_{\text{de}}$ starting at (p,q,k) on some $\alpha \in \Sigma^\omega$ is accepting, then for all $k \leq_{\checkmark} k'$ also the run of $\mathcal{A}_{\text{de}}$ 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 0.1.1. □

Lemma 0.1.3. *Let $\mathcal{A}$ be a DPA and $\rho \in Q_{\text{de}}^\omega$ 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 0.1.4. *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 $\mathcal{A}_{\text{de}}(p,q)$ 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 0.1.3, k can only become lower from there on. As $\leq_{\checkmark}$ is a well-ordering, a minimum must be reached at some point n_k.

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

Lemma 0.1.5. *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 0.1.4 (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 0.1.6. *Let $\mathcal{A}$ be a DPA. Then $\leq_{\mathrm{de}}$ is reflexive and transitive.*

Proof. For reflexivitiy, we need to show that $q \leq_{\mathrm{de}} q$ for all states q. This is rather easy to see. For a word $\alpha \in \Sigma^\omega$, the third component of states in the run of $\mathcal{A}_{\mathrm{de}}(q,q)$ on 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 0.1.4 (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 0.1.5 this gives us $\alpha \in L(\mathcal{A}_{\mathrm{de}}(q_1,q_3))$, letting us conclude in a contradiction.

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

First, to show $l_3 \leq l_2$, let $m \geq n_0$ be a position with $c(q_2(m)) = l_2$ and let $n \geq m$ be the minimal position with $k_{23}(n) = \checkmark$. If $m = n$, then $c(q_3(n)) \preceq_p c(q_2(n)) = l_2$ and therefore $c(q_3(n)) \leq l_2$. Otherwise, from m to $n-1$, k_{23} only grows smaller and is at most l_2 (Lemma 0.1.3). As the priority of q_2 never becomes an odd number smaller than l_2, the only way for $k_{23}(m)$ to be $\checkmark$ is that $c(q_3(m))$ is even and $c(q_3(m)) \leq k_{23}(m-1) \leq l_2$.

Second, assume that l_3 is odd and let m be a position with $c(q_3(m)) = l_3$. As l_2 is even, we have $k_{23}(m) \leq l_3 < l_2$. At no future position can $c(q_3)$ both be even and smaller than k_{23}, so k_{23} never becomes $\checkmark$ again. Thus, 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 0.1.7. *Let $\mathcal{A}$ be a DPA. Then $\equiv_{\mathrm{de}}$ 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 0.1.2, 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 0.1.8. *Let $\mathcal{A}$ be a DPA and $\equiv_{\mathrm{de}}$ the corresponding delayed simulation-relation. The quotient automaton $\mathcal{A} /_{\equiv_{\mathrm{de}}}$ 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 0.1.10 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 0.1.9. *Let $\mathcal{A}$ be a DPA with states $p, q \in Q$. Then $p \preceq_{\mathrm{de}} q$ 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 0.1.4, there is a position n such that k_i does not change anymore for $i \geq n$. We define $w = \alpha[0, n_0]$ and $\beta = \alpha[n_0 + 1,]$ (so $\alpha = w\beta$) and claim that w is a valid counterexample for the right-side property.

The first case is: $c(p') < c(q')$ and $c(p')$ is even.
Reading 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 $p \not\preceq_{\text{de}} q$.

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

Assume towards a contradiction that k_i does become $\checkmark$ again at some point. Let $j \geq |w|$ be the minimal position with $k_{j+1} = \checkmark$. Then by definition of 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 $\preceq_{\text{de}}$ seems arbitrary, it allows for an easier definition of $\equiv_{\text{de}}$ as is seen in the following statement.

Corollary 0.1.10. *Let $\mathcal{A}$ be a DPA with states $p, q \in Q$. Then $p \equiv_{\text{de}} q$ if and only if the following holds for all words $w \in \Sigma^*$:*

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

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 0.1.11. *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, $\text{Inf}(c(\pi)) = \text{Inf}(c(\rho))$ and π is accepting iff ρ is accepting. □

Lemma 0.1.12. *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 $\pi(0) \equiv_{de} \rho(0)$, then $\min Occ(c(\pi)) = \min Occ(c(\rho))$.*

Proof. Let $k = \min \mathrm{Occ}(c(\pi))$ and $l = \min \mathrm{Occ}(c(\rho))$. 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 $\mathcal{A}_{\mathrm{de}}(\pi(0), \rho(0))$ 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 $c(\rho(n)) \not\trianglerighteq_p c(\pi(n))$. 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 $\mathcal{A}_{\mathrm{de}}(\rho(0), \pi(0))$. As soon as $c(\pi)$ reaches its minimum, the third component of the run will never change again. □

Theorem 0.1.13. *Let $\mathcal{A} = (Q, \Sigma, q_0, \delta, c)$ be a DPA and let $p, q \in Q$ with $p \equiv_{\mathrm{de}} q$ and $c(p) < c(q)$. Define $\mathcal{A}' = (Q, \Sigma, q_0, \delta, c')$ with $c'(s) = \begin{cases} c(p) & \text{if } s = q \\ c(s) & \text{else} \end{cases}$. 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 0.1.12, 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. □

Corollary 0.1.14. *For a DPA $\mathcal{A}$, the quotient automaton $\mathcal{A}/_{\equiv_{\mathrm{de}}}$ is a DPA that recognizes the same language.*

0.1.2 Using delayed simulation for APAs

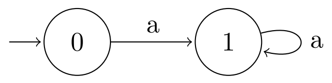

Figure 1: Example automaton in which the states could be merged but delayed simulation separates them.

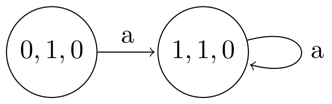

Figure 2: Example automaton in which the states could be merged but delayed simulation separates them.

0.1.3 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 1; 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 2.

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

Definition 0.1.4. 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 0.1.15. *$\equiv_{deR}$ is a congruence relation.*

Lemma 0.1.16. *Let $\mathcal{A}$ be a DPA with two states p and q. If $p \leq_{\mathrm{de}} q$, then $p \leq_{deR} q$.*

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

We know $k_0 = l_0$ by definition. For the sake of induction, we look at position $i+1$. If neither p_i nor q_i change their SCC in this step, the statement follows from lemma 0.1.1. Otherwise, $l_{i+1} = \checkmark \geq_{\checkmark} k_{i+1}$. □

The real difference comes up when looking at theorem 0.1.13. If we consider again the example given in figure 1, if that theorem would hold the same for $\equiv_{\text{deR}}$ we could assign both states the priority 0, which would then make the automaton accept every word. This is obviously not the same as the original automaton, so this statement deserves some additional inspection.

Lemma 0.1.17. *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 $k = \min \operatorname{Occ}(c(\pi[\text{off},\omega]))$ and $l = \min \operatorname{Occ}(c(\rho[\text{off},\omega]))$. 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 $\mathcal{A}_{\text{de}}(\pi(0),\rho(0))$ on α. Let $n \geq \text{off}$ be a position at which $c(\pi(n)) = k$. We claim that for all $i \geq n$, the third component of $\sigma(i)$ is k.

At $\sigma(n)$, this must be true because $k < l \leq c(\rho(n))$ and thus $c(\rho(n)) \not\leq_p c(\pi(n))$. 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 $\mathcal{A}_{\text{de}}(\rho(0),\pi(0))$. As soon as $c(\pi)$ reaches its minimum, the third component of the run will never change again. □

Theorem 0.1.18. *Let $\mathcal{A} = (Q,\Sigma,q_0,\delta,c)$ be a DPA and let $p,q \in Q$ with $p \equiv_{\text{de}} q$ and $c(p) < c(q)$. Define $\mathcal{A}' = (Q,\Sigma,q_0,\delta,c')$ 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 0.1.12, 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 0.1.19. *For a DPA $\mathcal{A}$, the quotient automaton $\mathcal{A}/_{\equiv_{\text{de}}}$ is a DPA that recognizes the same language.*

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 0.1.5. 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.

[] provides an algorithm that produces a normalized priority function when adapted slightly; see **??** for details.

Lemma 0.1.20. *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 $p \equiv_{\text{de}} q$ 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 0.1.10, 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 0.1.6. 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 0.1.7. Let $\mathcal{A}$ be a DPA with SCCs $\mathcal{S} \subseteq 2^Q$. Let $\preceq \in \mathcal{S} \times \mathcal{S}$ 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. $S_0 \prec S_1 \prec \cdots \prec S_{|\mathcal{S}|}$.

We inductively define a sequence of automata $(\mathcal{B}_i)_{0 \leq i \leq |\mathcal{S}|}$ and a sequence of relations $(R_i)_{0 \leq i \leq |\mathcal{S}|}$.

- The base case is $\mathcal{B}_{|\mathcal{S}|} = \mathcal{A} \restriction_{S_{|\mathcal{S}|}}$ and $R_{|\mathcal{S}|} = \{(p, q) \in S_{|\mathcal{S}|} \mid \forall w : c(\delta^*(p, w)) = c(\delta^*(q, w))\}$, which is the Moore equivalence of $\mathcal{A} \restriction_{S_{|\mathcal{S}|}}$.
- $\mathcal{B}_i$ prepends S_i to $\mathcal{B}_{i+1}$. We write $Q_i = \bigcup_{j=i}^{|\mathcal{S}|}$ for the state set of $\mathcal{B}_i$.

- Let R'_{i+1} be the transitive closure of $R_{i+1} \cup \{(p,q) \in Q_i \mid c(p) = c(q)\}$. Let R''_{i+1} be the congruence refinement of R'_{i+1} in $\mathcal{B}_i$. If S_i is a non-trivial SCC, then $R_i := R''_{i+1}$. Otherwise $S_i = \{q\}$. If q is not R''_{i+1}-equivalent to any other state but there is a $p \in Q_{i+1}$ such that for all $a \in \Sigma$, $(\delta(q,a), \delta(p,a)) \in R''_{i+1}$, then R_i is the transitive closure of $R''_{i+1} \cup \{(q,p),(p,q)\}$. Otherwise, $R_i := R''_{i+1}$.

We call R_0 the *iterated Moore equivalence*, $\sim_{IM}$, 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 0.1.21. *Let $\mathcal{A}$ be a DPA and let $\sim$ and $\sim'$ be equivalence relations. Let $\approx$ and $\approx'$ be the congruence refinements of $\sim$ and $\sim'$. If $\sim \subseteq \sim'$, then also $\approx \subseteq \approx'$.*

Proof. □

Lemma 0.1.22. *Let variables be defined as in definition 0.1.7 and let $\sim_M$ be the Moore equivalence of $\mathcal{A}$. Every R_i is a congruence relation on Q_i and if $(p,q) \in R_i$, then $p \sim_M q$.*

Proof. In addition to the statement of the lemma, we also show that every R'_i is an equivalence relation on Q_i, $(p,q) \in R'_i$ implies $c(p) = c(q)$, every R''_i is a congruence relation on Q_i, and $(p,q) \in R''_i$ also implies $p \sim_M q$.

By definition, $R_{|\mathcal{S}|}$ is the Moore equivalence of $\mathcal{B}_{|\mathcal{S}|}$ and therefore it is a congruence that implies Moore equivalence.

Now for the general case, assume that R_{i+1} was shown to be a congruence that implies Moore equivalence. R'_{i+1} is reflexive, as R_{i+1} is, and $c(q) = c(q)$ for every state $q \in S_i$. For the same reason, R'_{i+1} is symmetric and it is transitive by definition. Therefore, R'_{i+1} is an equivalence relation. The implication of priority equality by R'_{i+1} is clear from the definition.

R''_{i+1} is a congruence relation by definition. We know that $(p,q) \in R'_{i+1}$ implies $c(p) = c(q)$, so from lemma 0.1.21, the implication of Moore equivalence by R''_{i+1} follows.

If $R_i = R''_{i+1}$, the properties translate of course, so the only case left to consider is that R_i is transitive closure of $R''_{i+1} \cup \{(q_0,p_0),(p_0,q_0)\}$. The fact that R_i is an equivalence relation should be clear again, as well as the fact that $(p,q) \in R_i$ implies $p \sim q$, since $\sim_M \subseteq R''_{i+1} \subseteq R_i$.

Let $(p,q) \in R_i$ and $p' = \delta(p,a), q' = \delta(q,a)$ for some $a \in \Sigma$. We have to show that also $(p',q') \in R_i$. Since q_0 was not equivalent to any other states in R''_{i+1}, we either have $(p,q) \in R''_{i+1}$, or $q = q_0$ and $(p,p_0) \in R''_{i+1}$ (or the symmetric case). If $(p,q) \in R''_{i+1}$, then also $(p',q') \in R''_{i+1} \subseteq R_i$, as q_0 is not reachable from any state in Q_{i+1} by choice of $\preceq$.

Otherwise $q = q_0$ and $(p,p_0) \in R''_{i+1}$. Then $(\delta(p,a), \delta(p_0,a)) \in R''_{i+1}$ because it is a congruence relation and $(\delta(q_0,a), \delta(p_0,a)) \in R''_{i+1}$ by choice of p_0. Therefore, $(q',p') \in R''_{i+1} \subseteq R_i$. □

Lemma 0.1.23. *Let variables be defined as in definition 0.1.7. For every i, $L({}^{\mathcal{B}_i}/_{R_i}) = L(\mathcal{B})_i$.*

Proof. □

As $\mathcal{B}_0 = \mathcal{A}$ is a valid choice for the last automaton in the sequence, the corollary follows directly.

Corollary 0.1.24. *For a DPA $\mathcal{A}$, ${}^{\mathcal{A}}\big/_{\sim_{IM}}$ is also a DPA with $L({}^{\mathcal{A}}\big/_{\sim_{IM}}) = L(\mathcal{A})$.*

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

Theorem 0.1.25. *Let $\mathcal{A}$ be a DPA. Then $\equiv_{de} \subseteq \sim_{IM}$.*

Proof. $\square$

Computing a normalized priority function

If we can assure that our priority function is normalized, Moore-equivalence is a nice approximation of delayed simulation-equivalence. In fact, [] provides an algorithm that is suited for these needs. We will briefly reiterate that algorithm here, as the original paper puts the focus on a different aspect.

Algorithm 1 Normalizing the priority function of a DPA.

```
 1: function NORMALIZE($\mathcal{A}$)
Ensure: $c_N$ is normalized in $\mathcal{A}$.
 2:     $c_D \leftarrow$MAKEDENSE($\mathcal{A}$)
 3:     $c_N : Q \to \mathbb{N}, q \mapsto c_D(q)$
 4:     $Q' \leftarrow Q$
 5:     $k \leftarrow 0$
 6:     while $Q' \neq \emptyset$ do
 7:         $Q' \leftarrow Q' \setminus c_D^{-1}(k)$
 8:         $S \leftarrow$ Trivial SCCs of $\mathcal{A} \restriction_{Q'}$
 9:         for $q \in S$ do
10:             $c_N(q) \leftarrow k$
11:         end for
12:         $Q' \leftarrow Q' \setminus S$
13:         $k \leftarrow k + 1$
14:     end while
15:     return $c_N$
16: end function

17: function MAKEDENSE($\mathcal{A}$)
Ensure: For all $0 \leq i \leq \max c(Q)$, $c_D^{-1}(i) \neq \emptyset$.
18:     $c_D : Q \to \mathbb{N}, q \mapsto c(q)$
19:     for $i \leftarrow 0$ to $\max c(Q)$ do
20:         while $c_D^{-1}(i+1) = \emptyset$ do
21:             for $q \in Q : c_D(q) > i$ do
22:                 $c_D(q) \leftarrow c_D(q) - 2$
23:             end for
24:         end while
25:     end for
26:     return $c_D$
27: end function
```

Lemma 0.1.26. *The* NORMALIZE *function in algorithm 1 returns a normalized priority function for $\mathcal{A}$ that does not change the language.*

Proof. The fact that the language does not change after running NORMALIZE is part of the original paper.

To see that c_N is normalized, assume towards a contradiction that it is not. Let c □

Lemma 0.1.27. *The* NORMALIZE *function in algorithm 1 runs in $\mathcal{O}(nk)$, where n is the number of states and k is the number of priorities.*

Alternative delayed simulation

Algorithm 2 Compute $\equiv_{\text{de}}$ of a DPA $\mathcal{A}$.

```
function DelayedSimulation($\mathcal{A}$)
    $\mathcal{S} \leftarrow$ compute SCCs of $\mathcal{A}$
    $\mathcal{B} \leftarrow (\emptyset, \Sigma, 0, \emptyset, \emptyset)$
    while $\mathcal{S} \neq \emptyset$ do
        $s \in \mathcal{S}$ s.t. $s$ can reach no other SCC in $\mathcal{S}$
        $\mathcal{B} \leftarrow$ AddSCC($\mathcal{A}$, $\mathcal{B}$, $s$)
        $\mathcal{S} \leftarrow \mathcal{S} \setminus s$
    end while
    return $c_{\mathcal{B}}$
end function

function AddSCC($\mathcal{A}$, $\mathcal{B}$, $s$)
    $q_0 \in s$ arbitrary
    Add $s$ to $\mathcal{B}$: $\mathcal{B} \leftarrow (Q \cup s, \Sigma, q_0, \delta_{\mathcal{B}} \cup \delta_{\mathcal{A}} \upharpoonright_{s \times \Sigma}, c_{\mathcal{B}} \cup c_{\mathcal{A}} \upharpoonright_s)$
    $\sim_M \leftarrow$ Moore-equivalence of $\mathcal{B}$
    $\mathcal{C} \leftarrow \mathcal{B}/\sim_M$
    if $q_0$ is a trivial SCC in $\mathcal{A}$ and is not $\sim_M$-equivalent to any other states then
        if There is a state $p \in \mathcal{C}$ such that $\forall a : \delta_{\mathcal{B}}(q_0, a) = \delta_{\mathcal{B}}(p, a)$ then
            $c_{\mathcal{B}}(q_0) \leftarrow c_{\mathcal{C}}(p)$
        end if
    end if
    return $\mathcal{B}$
end function
```

Theorem 0.1.28. *Let $\mathcal{A} = (Q, \Sigma, q_0, \delta, c)$ be a DPA s.t. c is normalized and $\mathcal{A}$ contains no unreachable states. Let $c' =$ DELAYEDSIMULATION$(\mathcal{A})$. Then The delayed simulation-equivalence of $\mathcal{A}$ is the same as the Moore-equivalence of $(Q, \Sigma, q_0, \delta, c')$.*

Proof. Let $s_0, \dots, s_m$ be the sequence of SCCs as they are chosen in line 5 of the algorithm and let $\mathcal{B}_0, \dots, \mathcal{B}_{m+1}$ be the sequence of automata that are set in lines 3 and 6. The computation of $\equiv_{\text{de}}$ and $\sim_M$ is independent of the initial state of an automaton and the state set and transition function of $\mathcal{A}$ and $\mathcal{B}_{m+1}$ are the same. It thus suffices to show that for every $0 \leq i \leq m+1$, the Moore-equivalence of $\mathcal{B}_i$ is the same as the delayed-simulation equivalence of $\mathcal{A} \upharpoonright_{Q(\mathcal{B}_i)}$.

For $i = 0$, this is clear, as $Q(\mathcal{B}_0) = \emptyset$. □