0.1 Schewe

0.1.1 Schewe Automaton

Definition 0.1.1. 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 0.1.2. 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 0.1.1. *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. □

0.1.2 Schewe automaton structure

Lemma 0.1.2. *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 0.1.3. *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 0.1.2, 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 0.1.4. *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 0.1.2, 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 0.1.5. *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. □

0.1.3 Special cases of the Schewe automaton

Definition 0.1.3. 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 0.1.6. *Priority almost-equivalence is a congruence relation.*

Proof. Obvious. □

Lemma 0.1.7. *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 0.1.8. *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 0.1.4, 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 0.1.5, 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 0.1.9. *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 0.1.3, 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 0.1.2, 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 0.1.4. 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 0.1.10. *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 0.1.11. *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 0.1.10, 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 0.1.1, 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. □

0.1.4 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 0.1.5. 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 0.1.12. *For a given DPA $\mathcal{A}$, the Moore-minimization can be computed in $\mathcal{O}$.*